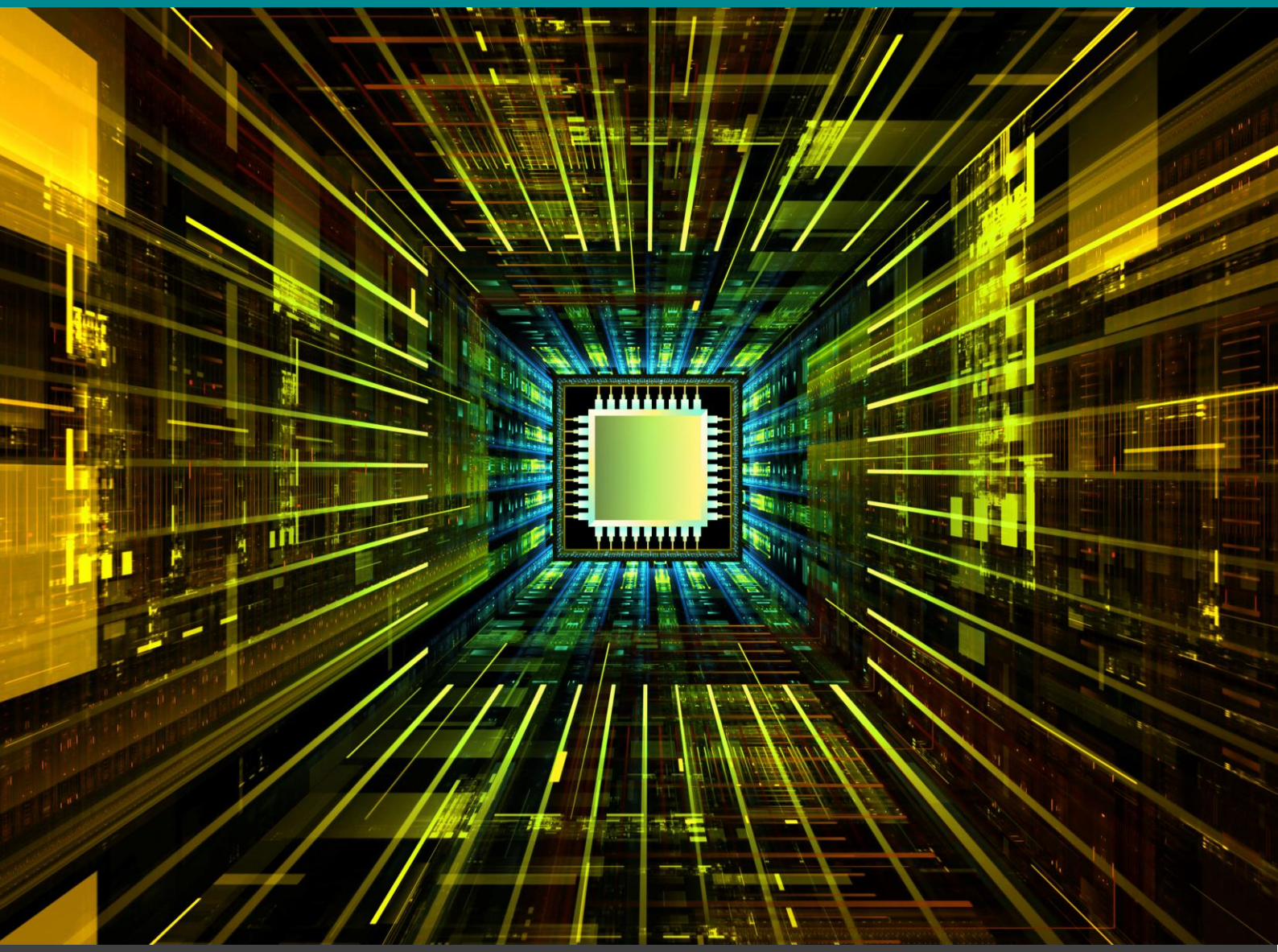
Juillet 2023
Smals Research

Smals
ICT for society



Informatique confidentielle

État de l'art

Fabien A. P. Petitcolas, PhD

Table des matières

Abbréviations

Les abréviations dans le document sont faites à partir du terme anglais.

BIOS	Basic Input/Output System	Système élémentaire d'entrée/sortie
CRTM	Core Root of Trust for Measurement	Racine principale de confiance pour la mesure
EK	Endorsement Key	Clé d'approbation
FHE	Fully Homomorphic Encryption	Chiffrement totalement homomorphe
HE	Homomorphic Encryption	Chiffrement homomorphe
HSM	Hardware Security Module	Boîte noire transactionnelle
MPC	Secure Multiparty Computation	Calcul multipartite sécurisé
PHE	Partial Homomorphic Encryption	Chiffrement homomorphe partiel
REE	Regular Execution Environement	Environnement d'exécution régulier
RoT	Root of Trust	Racine de confiance
SEE	Secure Execution Environement	Environnement d'exécution sécurisé
SWHE	Somewhat Homomorphic Encryption	Chiffrement presque homomorphe
TA	Trusted Application	Application de confiance
TCB	Trusted Computing Base	Base informatique de confiance
TEE	Trusted Execution Environement	Environnement d'exécution de confiance
TPM	Trusted Platform Module	Module de plate-forme de confiance

On considère généralement que les données peuvent être dans trois états. Celles stockées, par exemple sur un disque dur ou dans une base de données, sont dites « *au repos* », celles envoyées d'un ordinateur à un autre, par exemple via un réseau, sont « *en mouvement*, » et les données traitées par le micro-processeur sont « *en cours d'utilisation* ».

Aujourd'hui, des primitives cryptographiques sont largement déployées pour protéger les données dans les deux premiers états, assurant intégrité et confidentialité. Cependant, le renforcement des réglementations sur la protection des données et de la vie privée et l'augmentation des cybermenaces de longue durée a fait naître le besoin de protéger les données pendant le calcul.

Malheureusement cette protection des données en cours d'utilisation reste difficile, en particulier dans le contexte d'infrastructures informatiques publiques partagées : il s'agit en effet d'exécuter un logiciel avec certaines garanties de sécurité sur un ordinateur distant qui peut être détenu et géré par une partie non fiable voire malintentionnée.

De nombreuses attaques

Les attaques possibles sont nombreuses.

Au **niveau logiciel** on compte toutes les attaques permettant de compromettre l'intégrité du système, que ce soit le système d'exploitation, l'hyperviseur ou le micrologiciel (« *firmware* »). Celles-ci sont rendues possibles via divers types de rançongiciels ou d'outils de piratage, via des pilotes de périphériques compromis, via l'exploitation d'une vulnérabilité du jour zéro, de l'injection d'erreurs logiques, ou encore, à cause d'employés malveillants ou de mauvaises pratiques de fabrication, etc. [1]–[8]. On compte également l'accès aux données à l'aide de canaux auxiliaires (« *side-channel attacks* »), ou celui dû à des erreurs de configuration ou de paramétrage du système. Enfin, plus rares, les attaques cryptographiques, suite à une analyse poussée d'un algorithme ou à l'exploitation d'une percée mathématique sont possibles.

Au **niveau physique**, on peut envisager une administratrice système ou un prestataire externe accédant sans autorisation au matériel, pouvant alors se connecter directement au circuit imprimé et copier des informations par accès direct à la mémoire vive (DIMM DMA), espionner la connexion à débit de données double (DDR), surveiller le bus informatique, etc.

Techniques principales

L'informatique confidentielle (« *confidential computing* ») apporte une solution technique à ces problèmes à travers trois méthodes principales :

- Le **chiffrement homomorphe** (« *homomorphic encryption* » – HE), partiel ou total, concerne le calcul direct sur des données chiffrées [9] et peut aider dans certaines classes de calculs, mais engendre un coût de par la surcharge de calcul [10].
- Le **calcul multipartite sécurisé** [11] (« *secure multiparty computation* » - MPC) est une classe de méthodes cryptographiques qui permet à des agents mutuellement méfiants de calculer conjointement des fonctions sur leurs données d'entrée sans révéler celles-ci à d'autres agents[1].
- Les **environnements d'exécution de confiance** (« *trusted execution environment* » – TEE) s'appuient sur une combinaison de composants matériels sécurisés, de mécanismes d'attestation et de logiciels conçus pour utiliser ces fonctionnalités matérielles[2].

Dans la suite de ce document nous expliquons plus en détail le fonctionnement de ces trois méthodes. Bien entendu, il est également possible de les combiner, comme par exemple les environnements d'exécution de confiance avec le chiffrement homomorphe (par ex. [14]).

Nous faisons remarquer que certains auteurs ont proposé des méthodes d'externalisation des calculs basées sur des **fonctions pseudo-aléatoires** (par ex. [15]) ou encore sur la **perturbation des données** (par ex. ajout de bruit comme pour les techniques de « *differential privacy*[3] », remplacement des données par d'autres avec une statistique équivalente), mais celles-ci ont des applications limitées. Une étude sur l'ensemble de ces différentes techniques est fournie dans [16].

[1] https://www.smalsresearch.be/secure-multiparty-computation-collectieve-berekeningen-op-verspreide-gevoelige-gegevens/

[2] Plusieurs définitions des environnements d'exécution de confiance (TEE) ou des environnements d'exécution sécurisés (SEE) ont été données, parfois avec des contradictions entre elles. Dans ce document nous nous basons principalement sur les spécifications du *Confidential Computing Consortium* et de la GlobalPlatform [12]. Différentes définitions sont comparées et une description formelle de ces environnements est proposée dans [13].

[3] https://www.smalsresearch.be/differential-privacy/

1 Chiffrement homomorphe

Le chiffrement homomorphe[4] (HE) est une construction cryptographique qui consiste à calculer directement sur des données chiffrées de sorte que le déchiffrement donne le résultat du calcul correct sous forme de texte brut. Par exemple, supposons que les variables $m_1 = 5$ et $m_2 = 7$ soient chiffrées en c_1 et c_2 respectivement avec un système de HE approprié : une machine peut calculer directement $c_1 \boxplus c_2$[5], qui peut être déchiffré avec la clé secrète sk, pour donner le résultat $Dec(c_1 \boxplus c_2, sk) = 12$. De manière très concrète et simplifiée, l'idée est similaire à l'utilisation du logarithme pour les calculs de multiplications comme le faisaient nos grands-parents : si $v_1 = 35$ et $v_2 = 17$ et que l'on souhaite calculer $v_1 \times v_2$ on transforme d'abord v_1 et v_2 en $l_1 = \log_{10}(v_1) = 1{,}544068$ et $l_2 = \log_{10}(v_2) = 1{,}230449$ à l'aide d'une table de logarithmes, puis on calcule $l_1 + l_2 = 2{,}774817$. On peut demander à une autre personne de calculer pour nous la somme des logarithmes. Pour connaître le résultat final on utilise la table en sens inverse pour $v_1 \times v_2 = 10^{l_1+l_2} = 595$. Si l'autre personne n'a pas de table de logarithmes à sa disposition, il lui sera difficile de deviner ce résultat.

Dans un cadre basé sur une infrastructure informatique publique, cela permet à un client de chiffrer des données sensibles, et de les transmettre à une application résidant sur l'infrastructure publique afin d'effectuer des opérations de calcul intensif, qui seront ensuite retournées au client pour déchiffrement (Figure 1).

Le HE se présente généralement sous la forme de système de chiffrement à clé publique, où les données sont chiffrées à l'aide d'une clé publique, calculées, puis déchiffrées à l'aide de la clé privée correspondante. Certains schémas de HE symétriques (même clé de chiffrement et de déchiffrement) ont également été proposés dans la littérature académique.

Le type, le nombre et la performance des opérations permises par le HE varie selon les schémas de chiffrement. Certains schémas, comme celui de Paillier [17] ne permettent d'utiliser que les additions dans le domaine chiffré, tandis que d'autres comme le HE basé sur RSA [18] permettent seulement les multiplications des chiffrés. Ces schémas sont appelés « chiffrement homomorphe partiel » (PHE). Une autre famille d'algorithmes permet un plus grand nombre d'opérations, mais n'autorise de les appliquer qu'un nombre limité de fois avant que les informations chiffrées deviennent irrécupérables. Ces schémas sont qualifiés de « presque homomorphes » (SWHE). Enfin, il existe des méthodes de chiffrement « totalement homomorphes » (FHE)[6] qui permettent l'exécution de diverses opérations de façon illimitée, mais au prix d'une augmentation significative du temps de calcul et de l'espace de stockage requis (par ex. voir note de bas de page 49) [20], [21].

Les sections suivantes décrivent plus en détails ces différents types de chiffrements homomorphes.

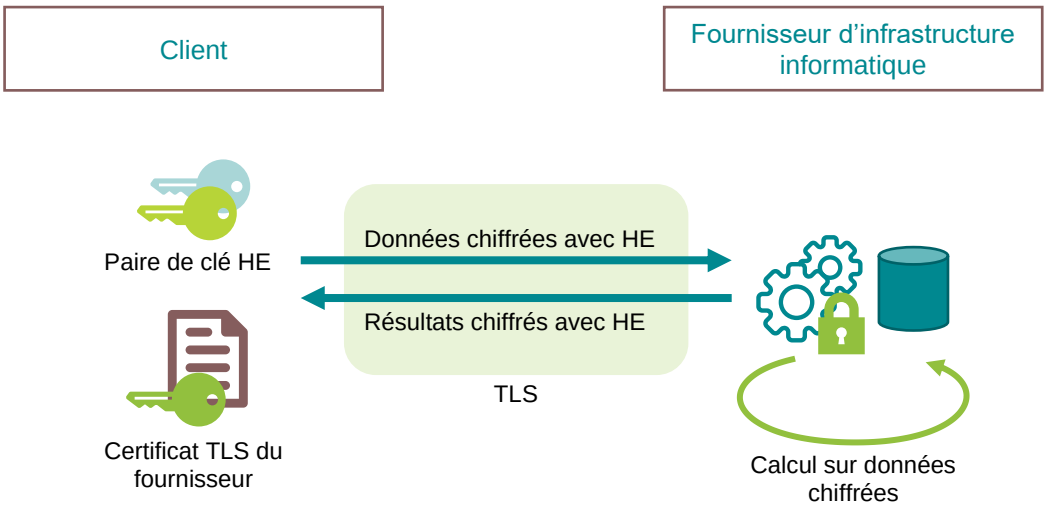


FIGURE 1 – SCHÉMA NOTIONNEL D'UTILISATION DU CHIFFREMENT HOMOMORPHE.

[4] Du grecque *ὁμός* (homos) signifiant « même » et *μορφή* (morphe) signifiant « forme. »

[5] Le symbole ⊞ désigne des additions homomorphes.

[6] Pour le FHE, le manque de standardisation des paramètres et des schémas a probablement contribué à sa mauvaise adoption. Le consortium *Homomorphic Encryption Standardisation* a été créé pour résoudre ce problème [19].

1.1 Chiffrement homomorphe partiel

Le HE a été découvert pour la première fois en 1978, lorsque Rivest et al. ont montré que la primitive du système de chiffrement RSA prend en charge la multiplication des textes chiffrés [18]. Étant donné un message m_1 chiffré avec le système RSA utilisant une clé publique de la forme (e, n), c'est-à-dire $E(m_1) = m_1^e \bmod n$, il est alors possible de chiffrer un second message $E(m_2) = m_2^e \bmod n$ et de multiplier les messages chiffrés afin d'obtenir $m_1 \cdot m_2$, grâce à la propriété suivante :

$$\begin{aligned} E(m_1) \cdot E(m_2) &= m_1^e \cdot m_2^e \bmod n \\ &= (m_1 \cdot m_2)^e \bmod n \\ &= E(m_1 \cdot m_2) \end{aligned}$$

Le client peut alors déchiffrer $c = E(m_1 \cdot m_2)$ avec sa clé de déchiffrement afin de trouver le résultat $m_1 \cdot m_2$.

Il faut cependant remarquer que cette utilisation de la propriété d'homomorphisme du chiffrement RSA doit s'appliquer sans remplissage (« *padding* »), rendant alors le chiffrement vulnérable à des attaques. Afin de pallier ce problème, RSA est généralement utilisé en conjonction avec un système de remplissage standard comme dans la famille PKCS #1 mais avec le désavantage d'annuler les propriétés d'homomorphisme.

1.1.1 Méthode de Goldwasser-Micali

La méthode de chiffrement de Goldwasser-Micali [22] proposée en 1982, permet de calculer la somme binaire (XOR) de deux messages dans le domaine chiffré. Ce schéma impose d'encoder les messages sous forme de séquences de bits d'information qui sont ensuite chiffrés bit par bit. Les données transmises sont alors la concaténation de ces bits chiffrés.

La sécurité de ce schéma est basée sur la même supposition qu'il est impossible pour un ordinateur de factoriser en un temps raisonnable le produit $N = p \cdot q$ de deux très grands nombres premiers p et q.

Malheureusement, un défaut important est que le chiffrement de chaque bit du message demande un calcul modulo N qui, pour le moment, doit être au moins de 2048 bits, mais devra être d'au moins 3072 bits en 2024, selon les recommandations du BSI[7], afin d'empêcher les attaques par factorisation. Cela introduit un coût majeur en termes de temps de calcul et de taille des données transmises (le message chiffré est environ 2048 fois plus grand que le message original).

1.1.2 Schéma de Paillier

Contrairement aux deux précédents, le schéma de chiffrement de Paillier [17] permet un homomorphisme d'addition : deux textes chiffrés peuvent être ajoutés puis déchiffrés afin de produire l'addition des deux textes originaux. C'est l'un des schémas de chiffrement homomorphe les plus simples et nous en décrivons ici le fonctionnement pour les lecteurs les plus curieux.

Dans le schéma de Paillier, chaque partie communicante partage une paire de clés en choisissant aléatoirement deux nombres premiers très grand p et q et en calculant le produit $N = p \cdot q$. La clé publique est simplement $pk \coloneqq N$ tandis que la clé privée est $sk \coloneqq \{p, q\}$. L'espace des messages est $\mathcal{M} = \mathbb{Z}/N\mathbb{Z}$.

Pour chiffrer un message $m \in \mathcal{M}$, en utilisant la clé publique pk, il faut d'abord tirer aléatoirement un nombre $r \xleftarrow{R} [1, N)$ puis calculer le texte chiffré de la manière suivante :

$$\mathfrak{m} \coloneqq [\![m]\!] = (1 + mN) r^N \bmod N^2.$$

Afin de déchiffrer le texte chiffré $\mathfrak{m}$, le destinataire doit d'abord calculer r à partir de $\mathfrak{m}$ et de la clé secrète sk comme suit :

$$r = \mathfrak{m}^{N-1 \bmod (p-1)(q-1)} \bmod N$$

et ensuite retrouver le texte de départ avec :

$$m = \frac{(\mathfrak{m} r^{-N} \bmod N^2) - 1}{N}.$$

Les propriétés d'homomorphisme sont les suivantes :

$$\begin{cases} E(m_1 + m_2) &= E(m_1) \cdot E(m_2) \bmod N^2 \\ E(m_1 - m_2) &= E(m_1) / E(m_2) \bmod N^2 \\ E(a \cdot m) &= E(m)^a \bmod N^2 \end{cases}$$

On vérifie la première ainsi[8] :

$$\begin{aligned} E(m_1, r_1) &= (1 + m_1 N)\, r_1^N \bmod N^2 \\ E(m_2, r_2) &= (1 + m_2 N)\, r_2^N \bmod N^2 \end{aligned}$$

$$E(m_1, r_1) \cdot E(m_2, r_2)$$

[7] « *Bundesamt für Sicherheit in der Informationstechnik*. »

[8] La vérification des deux autres est laissée comme exercice pour le lecteur.

$$= (1 + m_1 N)\, r_1^N \cdot (1 + m_2 N)\, r_2^N \bmod N^2$$
$$= (1 + (m_1 + m_2)N) \cdot (r_1 \cdot r_2)^N \bmod N^2$$
$$= E(m_1 + m_2) \bmod N^2$$

On remarquera que le schéma contient la possibilité de produire la multiplication de deux messages en élevant un texte chiffré à la puissance d'un autre message. Mais étant donné deux textes chiffrés, il n'existe pas de méthode connue pour produire la multiplication des deux textes originaux. Malgré ces limitations, ce schéma de chiffrement a été utilisé dans plusieurs systèmes de la littérature académique pour permettre des votes électroniques [23], des évaluations de modèles d'apprentissage automatique [24] ou encore des requêtes dans des bases de données chiffrées [25].

Il a été montré que les performances du schéma de chiffrement de Paillier peuvent atteindre le chiffrement de près de 3 millions de bits par seconde sur un ordinateur de bureau classique (Intel i7-4600U) [26].

1.2 Chiffrement presque homomorphe

Ce type de chiffrement permet d'effectuer un plus grand nombre d'opérations dans l'espace des chiffrés, mais des limites importantes concernant le type et le nombre d'opérations sont imposées. Dans certains schémas de chiffrement presque homomorphe, le nombre d'opérations augmente le « bruit » dans le texte chiffré, et après un certain nombre d'applications de l'opération, les textes chiffrés dévient de leur « vrai » valeur, rendant le résultat inutile.

1.3 Chiffrement entièrement homomorphe

La première percée significative dans la création d'un schéma de chiffrement homomorphe est due à Gentry [9], qui a présenté une construction théorique utilisant du chiffrement à base de treillis[9]. Les performances du schéma de départ ont été améliorées. Ici nous présentons la méthode de base[10].

Ce schéma est essentiellement basé sur un schéma de chiffrement presque homomorphe bruyant dont les textes chiffrés se dégradent progressivement après chaque opération. La principale idée de Gentry, cependant, a été d'exécuter de manière homomorphe le circuit de déchiffrement dans l'espace chiffré sur des textes chiffrés bruyants, ce qui permet de produire un nouveau texte chiffré avec un bruit réduit. Cette procédure d'amorçage (appelée « *bootstrapping* ») peut être effectuée de manière récursive pour permettre théoriquement un nombre illimité d'opérations.

Le concept consistant à construire d'abord un schéma de chiffrement presque homomorphe et à utiliser une procédure d'amorçage a influencé beaucoup d'autres schémas de chiffrement entièrement homomorphes. Cependant, en raison de leur coût en terme de performance, les schémas de chiffrement entièrement homomorphes de première génération sont généralement peu pratiques [21]. Une mise en œuvre du schéma de Gentry dans [27] a montré que la génération de clés prenait jusqu'à 2,5 heures selon les paramètres, avec une taille de clé publique allant jusqu'à 2,25 Go. La procédure d'amorçage est également connue pour être particulièrement complexe ; l'évaluation de circuits plus approfondis, tels que AES, à l'aide de l'amorçage s'est avérée prendre environ 36 heures [21].

Les travaux qui ont suivi ceux de Gentry se sont concentrés sur l'amélioration des exigences de stockage et de temps d'exécution. Par exemple, Coron et al. ont proposé un schéma de compression FHE pour réduire la taille des clés publiques de plus de 2 Go à environ 10 Mo. Le schéma Brakerski et al. (BGV) [28] tente de limiter le besoin d'amorçage en réduisant le bruit généré lors des opérations dans l'espace chiffré. Par la suite, HELib a été introduit par Halevi et al. [29], fournissant une mise en œuvre C++ optimisée et open-source du schéma BGV. Cela a permis de diviser par 12 la complexité temporelle de l'évaluation des circuits AES, qui est passée de 36 à 3 heures.

Les recherches plus récentes dans le domaine du chiffrement entièrement homomorphe visent à réduire sa complexité pour atteindre des niveaux proches de la pratique. Si les générations précédentes se sont concentrées sur la procédure d'amorçage et la réduction de sa complexité, qui est passée de plusieurs heures à quelques minutes pour les opérations et la génération de clés, elles étaient toujours limitées à l'évaluation de fonctions de profondeur limitée. Par exemple, le

[9] En mathématiques, les treillis sont des ensembles partiellement ordonnés.

[10] Plus de détails dans [20].

schéma TFHE, proposé par Chillotti et al. [30], introduit une nouvelle méthode de chiffrement entièrement homomorphe dans laquelle la procédure d'amorçage est réduite à seulement 13 millisecondes en utilisant une clé de 16 Mo sans réduire la sécurité.

1.4 Avantages et limites

L'avantage majeur des schémas de chiffrement homomorphe est qu'ils sont basés sur la difficulté à résoudre certains problèmes mathématiques bien étudiés depuis très longtemps. Ces schémas de chiffrement homomorphe fonctionnent généralement selon les mêmes hypothèses opérationnelles que les autres systèmes de chiffrement à clé publique, tels que RSA. Bien-entendu cela nécessite un environnement sécurisé pour le stockage des clés et l'exécution des procédures de chiffrement et de déchiffrement, ainsi que de bonnes pratiques générales de gestion des clés, telles que la rotation et la révocation des clés.

Parce qu'ils sont basés sur des treillis, certains schémas de chiffrement, comme celui de Gentry (voir 1.3) ont l'avantage d'être résistants aux attaques cryptographiques post-quantiques.

En outre, l'intégrité, l'authenticité et la disponibilité ne sont pas les principales préoccupations du chiffrement homomorphe ; il est donc nécessaire de l'utiliser en conjonction avec un protocole de sécurisation des échanges comme TLS, des signatures numériques et d'autres techniques afin de répondre à des préoccupations autres que la confidentialité, telles que la protection des données contre les modifications non autorisées pendant le transport et au repos.

Un autre problème réside dans le fait que le chiffrement homomorphe ne peut pas garantir que les opérations homomorphes exécutées sont celles prévues, c'est-à-dire que $c_1 + c_2$ est exécuté et non $c_1 - c_2$. Il est donc nécessaire d'utiliser d'autre méthodes, telles que les TEE (voir section 3) et les éléments sécurisés, en complément ; pour répondre à cette exigence.

Enfin pour pallier la lenteur relative des schémas de chiffrement homomorphe, plusieurs groupes de recherche travaillent sur l'élaboration de matériel spécialisé [21], mais selon certains auteurs, même avec l'accélération fournie par le matériel pour les FHE, ceux-ci restent peu pratiques à cause d'une latence importante et des approximations introduites lors des étapes successives de chiffrement et déchiffrement.

1.5 Bibliothèques existantes

Dans cette section, nous examinons les bibliothèques open-source populaires développées par la communauté scientifique pour effectuer un chiffrement homomorphe.

1.5.1 SEAL

La bibliothèque « *Simple Encrypted Arithmetic Library* » (SEAL) est une bibliothèque C++ développée par le groupe de recherche en cryptographie de Microsoft Research [31]. La bibliothèque est sous licence MIT et ne nécessite aucune dépendance externe supplémentaire. SEAL fournit une implémentation du schéma de Fan-Vercauteren [32]. La quantité de bruit autorisée, et donc d'opérations, est déterminée par un ensemble complexe de paramètres.

1.5.2 HELib

HELib[11] est une bibliothèque C++ sous licence Apache développée par Halevi et al. [29] chez IBM Research. Elle met en œuvre le schéma FHE BGV et dépend de la bibliothèque mathématique NTL[12]. Contrairement à SEAL, la bibliothèque prend entièrement en charge l'homomorphisme avec procédure d'amorçage ; cependant, les résultats indicatifs présentés dans [33] montrent que l'évaluation d'un circuit AES avec amorçage prend environ 15 minutes avec HELib.

1.5.3 TFHE

TFHE[13] est une bibliothèque C++ de bas niveau, sous licence Apache, permettant de mettre en œuvre le schéma TFHE [30]. Elle est développée conjointement par l'ÉPFL, l'Université de Versailles Saint Quentin en Yvelines, l'Université Paris-Saclay et la KU Leuven, et soutenue par Gemalto.

La bibliothèque permet d'évaluer des circuits booléens arbitraires, composés de portes binaires, dans un espace chiffré. La bibliothèque prend en charge l'évaluation de 10 portes logiques. Les auteurs déclarent que

[11] https://github.com/homenc/HElib
[12] https://libntl.org/
[13] https://www.tfhe.com/

« *l'évaluation de chaque porte logique prend environ 13 millisecondes en temps mono-cœur* » [34]. Comme pour les autres bibliothèques, un ensemble de paramètres du schéma doit être choisi, ce qui affecte les performances et la sécurité du schéma.

1.5.4 OpenFHE

OpenFHE[14] est une bibliothèque de chiffrement entièrement homomorphe écrite en C++. Elle inclut des mises en œuvre efficaces de la plupart des algorithmes communs de chiffrement entièrement homomorphe pour les entiers, les nombres réels et l'arithmétique booléenne ainsi que des extensions permettant le chiffrement homomorphe avec plusieurs agents [35].

1.6 Conclusion

Bénéficiant de fondements mathématiques solides, le chiffrement homomorphe permet d'effectuer des opérations sur des données sensibles tout en conservant leur confidentialité, mais, en soi, il n'assure pas l'intégrité des données, c'est-à-dire que celles-ci peuvent être modifiées lors du transport entre le client et le fournisseur d'infrastructure informatique par exemple. Il ne fournit pas non plus d'authentification – ce qui peut être réalisé en ajoutant des signatures numériques aux données chiffrées.

Étant donné qu'un seul agent est en possession de la clé de déchiffrement, les schémas de chiffrement homomorphe offrent moins de flexibilité lorsque que plusieurs agents souhaitent travailler sur les mêmes données, contrairement aux méthodes de calcul multipartite sécurisé auxquelles nous nous intéresserons dans la prochaine section.

Enfin les points faibles principaux de cette technique d'informatique confidentielle restent, malgré les améliorations importantes depuis les premiers schémas : la faible performance de calcul, la complexité du paramétrage de certaines bibliothèques et la nécessité de réécrire le code des applications existantes afin d'intégrer ces bibliothèques. Ces deux derniers points imposent le recours à des experts en cryptographie afin de fournir les recommandations nécessaires lors d'une mise en œuvre.

[14] https://github.com/openfheorg/openfhe-development

2 Calcul multipartite sécurisé

Dans la précédente section, nous avons vu comment un agent pouvait effectuer des opérations sur ses données en utilisant les ordinateurs d'un autre agent en utilisant un système de chiffrement particulier. Ici nous nous intéressons à un problème un peu plus général où plusieurs agents mutuellement méfiants souhaitent calculer conjointement des fonctions sur leurs données respectives sans révéler celles-ci aux autres agents et de telle façon que le résultat soit exact.

Une façon simple de résoudre ce problème est de faire appel à un tiers de confiance qui collecte les données de chaque agent, calcule le résultat et le renvoie aux agents via un canal sécurisé.

Une autre façon est d'utiliser un protocole de calcul multipartite sécurisée (« *secure multiparty computation* » – MPC). Un protocole MPC est un moyen d'effectuer un calcul sécurisé décentralisé entre des agents sans l'implication d'un tiers de confiance centralisé. Les agents font des calculs localement et s'envoient des messages selon certaines règles préétablies. Les agents peuvent utiliser le protocole MPC pour « mettre en commun » leurs données de manière sécurisée et calculer une fonction utile, par exemple calculer un maximum sur un ensemble, une somme, ou encore trouver des éléments communs à tous les agents (intersection d'ensemble), sans révéler les entrées en texte clair. Le résultat est ensuite renvoyé en texte clair aux agents participants qui n'apprendront rien de plus sur les entrées des autres agents que ce que révèle la fonction calculée.

L'idée de MPC a été introduite pour la première fois en 1982 par Yao [11] qui étudiait une solution au « problème des millionnaires » – un problème de cryptographie impliquant deux millionnaires qui souhaitent identifier le revenu le plus élevé sans divulguer leur richesse réelle. La solution, les circuits brouillés de Yao [11], a été conçue pour le cas bipartite, qui implique l'évaluation de circuits booléens chiffrés (voir [36] pour plus de détails).

2.1 MPC basé sur le partage de secrets

À la suite des travaux de Yao, d'autres ont généralisé le problème à plusieurs agents en utilisant le partage de clé secrète de Shamir, où les données sensibles sont divisées en un certain nombre de constituants, appelés « parts, » de sorte que chaque part ne révèle rien sur les données initiales, mais où plusieurs parts peuvent être combinées pour récupérer le secret d'origine (par ex. [37]). Ces parts ont des propriétés mathématiques

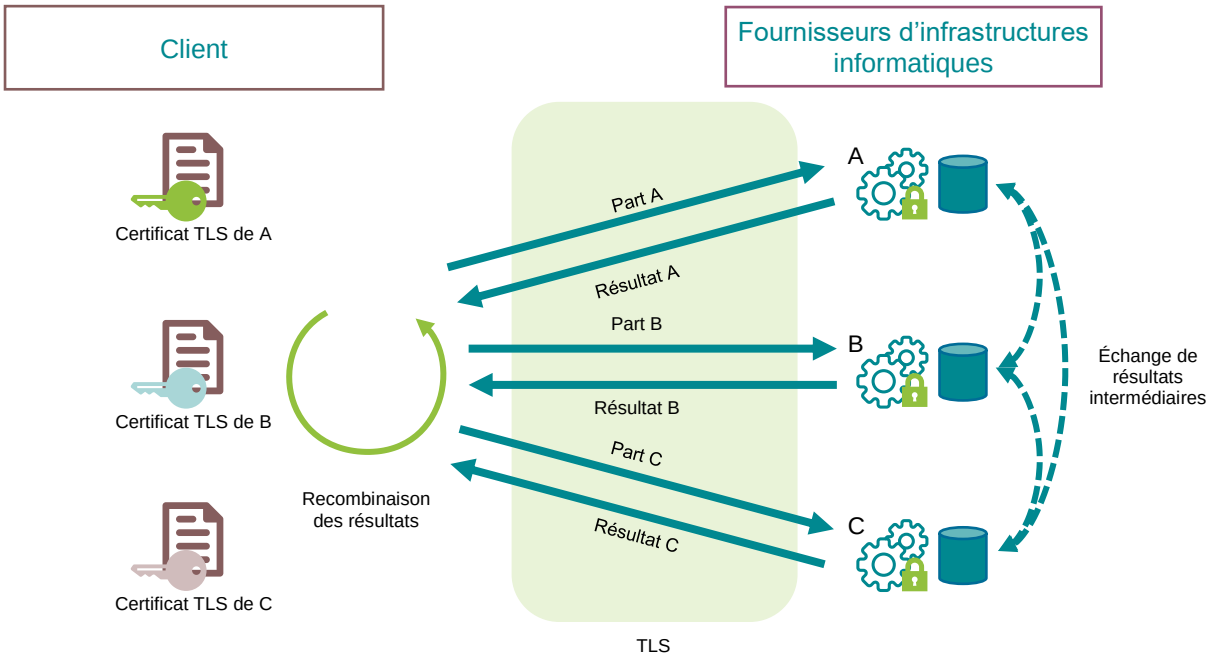


FIGURE 2 – MODÈLE DE DÉPLOIEMENT NOTIONNEL DANS LEQUEL UN CLIENT PARTAGE SES DONNÉES SENSIBLES ENTRE PLUSIEURS FOURNISSEURS D'INFRASTRUCTURE POUR EFFECTUER DES CALCULS PARTIELS, PUIS RECOMBINE LES RÉSULTATS LOCALEMENT.

intéressantes qui permettent des opérations arithmétiques. En transmettant ces parts à chaque partie, ces opérations peuvent être effectuées pour calculer une fonction d'intérêt. En général, le résultat des calculs est révélé à tous les agents.

En pratique, la fonction à calculer est d'abord décomposée en un circuit de calculs élémentaires (par ex. additions et multiplications) et chaque agent commence par partager les parts de sa propre valeur d'entrée avec les autres agents. Ensuite chaque agent exécute chaque étape du circuit, obtenant ainsi des valeurs intermédiaires, jusqu'à l'obtention du résultat final. Un schéma linéaire de partage de secret permet de calculer un résultat sans interaction avec les autres agents. En revanche une multiplication requiert une interaction plus ou moins complexe.

Les protocoles MPC peuvent également être utilisés pour partager les opérations sur des données sensibles entre plusieurs fournisseurs d'infrastructure informatique qui ne sont pas de connivence : tout d'abord le client peut partager ses données et envoyer chaque part à un fournisseur différent. Chaque fournisseur calcule alors un résultat partiel et le retourne au client, qui peut alors recombiner les résultats partiels pour obtenir le résultat final (Figure 2).

2.2 Modèle de menace

Ces schémas de partage peuvent tolérer des adversaires – actifs ou passifs – contrôlant un certain nombre d'agents :

- **Presque honnête** : l'adversaire exécute le protocole MPC comme il se doit, mais tente d'apprendre des informations à partir des autres sources, par exemple en observant la mémoire de la machine, et les données stockées (par ex. base de données, système de fichiers).
- **Malicieux** : en plus du comportement « presque honnête, » l'adversaire peux dévier de façon arbitraire du protocole en cours d'exécution. Ce cas s'applique lorsque l'adversaire peut contrôler l'exécution de l'application d'un des participants.

En général, les protocoles MPC sécurisés par rapport au second type d'adversaires sont moins efficaces que ceux ne s'appliquant qu'au premier.

De plus on attend qu'un nombre minimum d'agents suivent le protocole et, en particulier qu'ils ne partagent pas leurs « parts » avec d'autres agents et qu'ils fournissent des données d'entrée correctes. La sécurité d'un protocole MPC est donc souvent mesurée par le nombre d'agents corrompus pouvant être tolérés avant que le protocole échoue.

2.3 Limites

La principale limite du MPC est la surcharge en termes d'infrastructure nécessaire à l'orchestration des calculs conjoints et des coûts de communication importants imposés lors des opérations de calcul. Par exemple, chaque multiplication requiert un tour complet des parties, ce qui peut engendrer une latence non négligeable.

2.4 Bibliothèques existantes

La plupart des bibliothèques mettant en œuvre le MPC sont issues de projets universitaires et beaucoup ne sont plus activement développées. La documentation est souvent rare et largement incomplète, et la qualité du code manque souvent de maturité. En général ces bibliothèques permettent de s'abstraire de la gestion du réseau (connections, messages, calcul, reconstruction). En outre, certaines bibliothèques comme SPDZ et SCALE-MAMBA nécessitent l'utilisation d'un langage de programmation et d'un compilateur spécifique. Par conséquent, l'intégration de ces bibliothèques dans du code existant n'est pas triviale.

2.4.1 VIFF et MPyC

MPyC[15] est une bibliothèque Python sous licence MIT produite par TU Eindhoven et destinée au prototypage rapide de systèmes basés sur les MPC. MPyC repose en grande partie sur le « *Virtual Ideal Functionality Framework* » (VIFF) [37] de l'Université d'Aarhus, qui fournit une bibliothèque Python permettant la parallélisation automatique d'opérations de base, telle la multiplication, pour le MPC. En d'autres termes, MPyC peut être considéré comme un descendant direct de VIFF, dont le développement actif a cessé en 2013 ; MPyC est toujours en cours de développement actif.

[15] https://github.com/lschoe/mpyc

2.4.2 SPDZ et SCALE-MAMBA

SPDZ[16], de l'Université de Bristol, est un schéma MPC de sécurisé contre les adversaires malicieux, qui utilise le partage de secret en conjonction avec un chiffrement presque homomorphe ; il comprend une étape de pré-traitement hors ligne, où un aléa partagé est produit, avant une étape en ligne où le calcul de la fonction réelle est effectué. Le schéma SPDZ existe sous forme d'une bibliothèque open-source C++ ; cependant, le développement actif a cessé en 2018.

La KU Leuven propose un dérivé de SPDZ : également open-source et écrit en C++, SCALE-MAMBA[17] apporte diverses corrections et améliorations de performance. Il fournit une interface (« MAMBA ») qui compile un langage personnalisé de type Python en une représentation qui est exécutée sur la machine virtuelle SCALE pour effectuer le calcul sécurisé. Le système est relativement lent, complexe et gourmand en ressources.

2.5 Conclusion

Malgré quelques applications pratiques en particulier pour des collaborations spécifiques entre différentes entreprises ou institutions [38], le MPC reste encore à un stade très précoce, et, pour l'essentiel un domaine de recherche universitaire très actif où les connaissances évoluent rapidement.

La complexité de la mise en œuvre et la surcharge en termes de coût et de communication, le manque de prise en charge par des spécialistes de l'industrie, la nécessité d'utiliser des logiciels client et serveur très spécialisés, expliquent la faible maturité du calcul multipartite sécurisé.

[16] https://github.com/bristolcrypto/SPDZ-2

[17] https://github.com/KULeuven-COSIC/SCALE-MAMBA

3 Environnements d'exécution de confiance

Dans les deux précédentes sections, nous avons présenté deux techniques d'informatique confidentielle basées sur des méthodes mathématiques et cryptographiques. Bien qu'elles apportent une réponse intellectuellement satisfaisante, elles s'avèrent encore lourdes et complexes. Nous allons donc étudier ici une troisième méthode, beaucoup plus répandue, qui repose sur la mise en œuvre correcte du matériel et du micrologiciel constituant une base informatique de confiance (TCB).

En règle générale, la pile complète d'une plate-forme informatique est composée de plusieurs techniques provenant de nombreux fournisseurs différents, ce qui entraîne un manque de mise en œuvre cohérente des contrôles de sécurité. Cependant, la couche physique matérielle sur laquelle l'ensemble du système est construit reste la première couche de protection. Par conséquent une façon d'améliorer la sécurité globale d'un système – y compris en empêchant l'accès aux données – est d'augmenter la sécurité de la partie physique.

C'est sur cet argument que se basent les environnements d'exécution de confiance (TEE) qui visent à protéger l'exécution du code des applications contre les attaques de code privilégié et certaines attaques physiques en réduisant la base informatique de confiance et la surface d'exposition aux attaques (Figure 3).

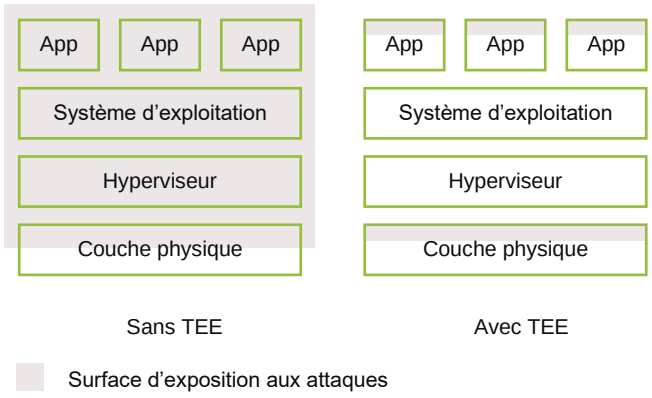


FIGURE 3 – SURFACE D'EXPOSITION AUX ATTAQUES TYPIQUE D'UN SYSTÈME SANS (À GAUCHE) ET AVEC TEE (À DROITE).

Dans certains cas, la petitesse du TEE permet l'application de méthodes formelles (basées sur des constructions mathématiques) et la vérification de la conformité de la mise en œuvre par rapport aux spécifications. Cela permet de renforcer le niveau de confiance placé dans le TEE.

Dans ce modèle d'utilisation, on envisage donc en général, les acteurs suivants :

- Le **fournisseur d'infrastructure** gère toute l'infrastructure informatique ; il a un accès physique à toutes les machines et contrôle tous les logiciels de la plate-forme, à l'exception des microprogrammes de confiance.
- Le **client** déploie des applications traitant des données sensibles sur des machines virtuelles hébergées par le fournisseur d'infrastructure informatique.
- Le **fabricant de matériel** est considéré comme une entité de confiance qui fournit le matériel (par ex., les microprocesseurs) et les micrologiciels de confiance.
- Enfin, l'**adversaire** cherche à rompre la confidentialité, l'intégrité et, dans une certaine mesure, la disponibilité des données du client. Il peut être un autre client du fournisseur d'infrastructure utilisant la même plate-forme que le client cible, ou simplement le fournisseur d'infrastructure lui-même.

3.1 Caractéristiques des TEE

Les TEE sont des systèmes de sécurité qui s'exécutent parallèlement aux environnements d'exécution réguliers (REE) (y compris le système d'exploitation comme Linux ou Windows). Ils fournissent une zone sûre pour protéger les actifs (par ex., les secrets sensibles, les chaînes d'authentification) et ont également pour but d'assurer que les opérations effectuées en leur sein et les données associées ne peuvent pas être visualisées à l'extérieur, pas même par un logiciel privilégié (par ex., hyperviseur, système d'exploitation) ou un débogueur. Combinés à un mécanisme d'attestation (voir 3.5) ils assurent que le code qu'ils exécutent ne peut pas être modifié ou remplacé sans l'autorisation du client.

Un TEE peut être utilisé pour fournir une isolation de mémoire, empêchant les applications partageant le même matériel sous-jacent de lire la mémoire allouée aux autres. Certains TEE permettent d'isoler des applications entières dans leurs propres enclaves plutôt que de protéger uniquement des opérations ou de la mémoire spécifique.

Les TEE sont conçus pour résister à un large éventail de menaces logicielles du système d'exploitation standard, y compris les menaces au niveau du noyau (par ex. [39]) provenant d'un périphérique « rooté » ainsi qu'à certaines attaques physiques. Les TEE font l'objet d'une analyse de sécurité rigoureuse dans le cadre des Critères communs [40], [41], mais ils manquent encore généralement de défenses contre les attaques matérielles complexes. Par exemple, les attaques qui « *nécessitent généralement un accès à long terme et/ou invasif au matériel, y compris les techniques de grattage de puces et les sondes de microscope électronique* » sont explicitement hors du champ d'application des exigences du *Confidential Computing Consortium* [42].

Les exigences de sécurité d'un TEE se résument ainsi (voir [12] pour plus de détails) :

- L'objectif principal d'un TEE est de **protéger ses actifs** contre les attaques à travers le reste de l'environnement d'exécution (voir 3.8.1). Ceci est réalisé grâce à des mécanismes matériels que ces autres environnements ne peuvent pas contrôler.
- Le TEE offre une **protection contre certaines attaques physiques** (voir 3.8.2) mais les attaques intrusives qui brisent physiquement la structure physique du circuit intégré ne sont pas considérées. Le profil de protection du TEE[18] propose une façon d'évaluer l'effort requis pour différentes attaques et notamment les attaques par canaux auxiliaires [41] (voir 3.8.2.2).
- Les composants du système (tels que les interfaces de débogage) capables d'accéder aux actifs du TEE sont désactivés ou contrôlés par un élément qui est lui-même un élément protégé de ce TEE.
- L'environnement d'exécution du système d'exploitation de confiance est démarré à partir d'une **racine de confiance** (RoT) à l'intérieur du TEE via un processus de démarrage sécurisé.
- Le TEE fournit un **stockage sécurisé** des données et des clés cryptographiques. Ce stockage est lié à ce TEE particulier sur un appareil particulier.
- Les logiciels extérieurs au TEE ne peuvent pas appeler directement les fonctionnalités exposées par les interfaces internes du TEE et doivent passer par des **protocoles de communication spécifiques** permettant au système d'exploitation de confiance de vérifier l'acceptabilité de l'opération non-TEE demandée.

Tous les TEE ne sont pas identiques et les fabricants peuvent proposer différentes mises en œuvre de TEE avec différentes propriétés de sécurité, différentes fonctionnalités et différents mécanismes de contrôle pour fonctionner sur les applications sécurisées[19]. En effet certains TEE s'adressent plutôt à des téléphones (par ex. technique TrustZone[20] d'ARM) tandis que d'autres à des ordinateurs de bureau ou des ordinateurs formant une infrastructure informatique publique (par ex. technique SGX[21] d'Intel ou SEV-SNP[22] d'AMD).

3.2 Vérification de l'intégrité

Un concept clé des TEE est la capacité du client à vérifier l'intégrité de la plate-forme sous-jacente. Cela est généralement réalisé grâce à deux composantes : la vérification du matériel avec ses différents modules logiciels, et la vérification des micrologiciels et logiciels de plus haut niveau. Dans le cadre d'infrastructures informatiques publiques, il est nécessaire de pouvoir effectuer cette vérification à distance (voir 3.5).

Les hachages cryptographiques (également souvent appelés « mesures » dans le contexte des TEE) du logiciel, du micrologiciel et des fichiers de configuration associés sous-tendent la création d'une chaîne de confiance. Chacune de ces chaînes de confiance commence par un module racine de confiance qui doit être aussi petit que possible. Il peut prendre la forme d'un bloc de démarrage implicitement fiable (par ex., stocké en mémoire à lecture seule) du système élémentaire d'entrée/sortie (BIOS) qui mesure d'abord sa propre in-

[18] Celui-ci a été approuvé par l'Agence nationale de la sécurité des systèmes d'information (ANSSI) de la France.

[19] Il y a des propositions pour simplifier le travail des programmeurs d'applications sécurisées avec, par exemple, un protocole interopérable de gestion de telles applications fonctionnant sur différents TEE [43].

[20] https://www.arm.com/technologies/trustzone-for-cortex-a

[21] https://www.intel.com/content/www/us/en/architecture-and-technology/software-guard-extensions.html

[22] https://developer.amd.com/sev/

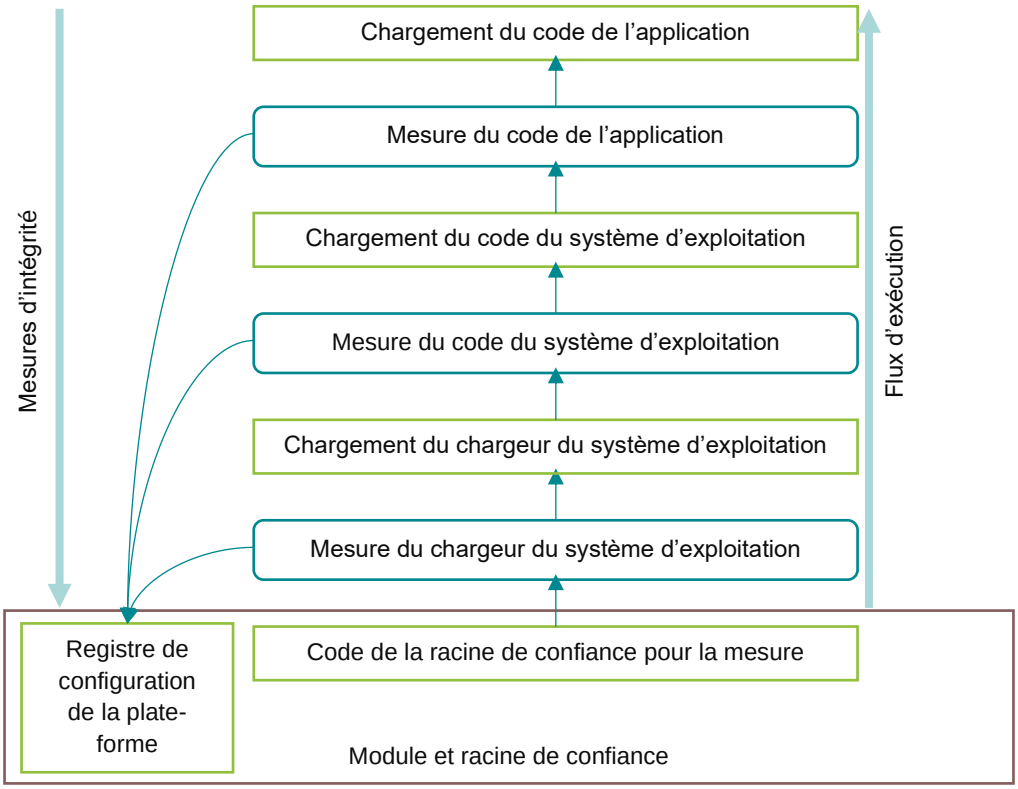


FIGURE 4 – SÉQUENCE DE DÉMARRAGE D'UNE PLATE-FORME SÉCURISÉE (ADAPTÉ DE [24]).

tégrité, puis étend la mesure à l'ensemble du BIOS. Ensuite, chaque micrologiciel ou logiciel supplémentaire est d'abord mesuré avant d'être exécuté et cela va jusqu'au système d'exploitation en suivant un processus de démarrage classique [44].

Ce mécanisme de *chaîne de confiance* résumé dans la Figure 4, fournit une vue fiable de l'état du système au moment de la mesure. Toute bibliothèque, module, application, partie du système d'exploitation ou non, a une mesure d'intégrité stockée dans un registre matériel sécurisé. En principe ces mesures sont signées afin d'empêcher les attaques par relecture ou par l'homme du milieu (par ex. [45]).

Les protections matérielles garantissent en principe qu'une fois une enclave lancée, aucune modification de son code ou de ses données statiques ne peut être effectuée depuis l'extérieur de l'enclave. Une fois lancés, les processus extérieurs à l'enclave peuvent recevoir une attestation cryptographique qui inclut une signature du code à l'intérieur de l'enclave, de sorte que le processus extérieur peut être assuré de l'exactitude de tout le code pouvant s'exécuter dans l'enclave. Cette attestation locale ou à distance implique la génération d'une signature des mesures, et en particulier celle de la racine principale de confiance pour la mesure[23] (CRTM) avec des clés d'attestation spécifiques.

Il convient de souligner que dans une architecture moderne complexe, le fabricant du CRTM peut ne pas rester longtemps l'autorité de confiance dans le processus de démarrage et il faut vérifier la chaîne de valeurs de hachage enregistrées pour déterminer l'autorité de l'état actuel du système. En outre, dans une infrastructure informatique typique, il peut y avoir des milliers de machines avec leur propre CRTM, ce qui rend leur suivi une tâche complexe, de sorte que des services d'attestation à distance dédiés sont souvent mis à la disposition du client de l'infrastructure informatique.

Notons enfin qu'une construction de racine de confiance avec des protections matérielles physiques sera plus difficile à changer qu'une construction uniquement dans le micrologiciel. Mais un matériel immuable de racine de confiance ne peut pas être mis à jour.

[23] Le « *Core Root of Trust for Measurement* » est le tout premier code du BIOS qui est exécuté dans le microprocesseur principal au moment de l'allumage. Il est souvent stocké dans un module de plate-forme de confiance (voir 3.3).

3.3 Prise en charge matérielle

L'application des propriétés mentionnées ci-dessus est basée sur un matériel physique spécifique, en intégrant tout ou partie des ressources nécessaires au bon fonctionnement du TEE dans un même boîtier, ou en utilisant des ressources partagées par les autres composants disponibles sur le circuit imprimé de la machine (par ex. la mémoire vive). Dans ce dernier cas une isolation par chiffrement est nécessaire.

Les boîtes noires transactionnelles (« *hardware security module* » – HSM) sont des « *dispositifs informatiques physiques qui protègent et gèrent les clés cryptographiques et fournissent un traitement cryptographique* » [46] : cela inclut le chiffrement, le déchiffrement, la génération de signatures et la vérification des signatures.

Un type spécial de HSM sont les modules de plate-forme de confiance (« *trusted platform module* » – TPM) qui peuvent générer des clés cryptographiques et d'autres informations sensibles. Le *Trusted Computing Group* (TCG) définit des schémas pour établir la confiance dans une plate-forme ; ils sont basés sur l'identification de composants matériels et logiciels de cette plate-forme. « *Un TPM fournit des méthodes pour collecter et signaler ces identités. Un TPM utilisé dans un système informatique rend compte du matériel et des logiciels d'une manière qui permet de déterminer le comportement attendu et, à partir de cette attente, d'établir la confiance* » [47]. Il est conçu pour permettre à un agent indépendant de déterminer si la base informatique de confiance d'un système a été compromise. Certains TPM sont mis en œuvre à travers des composants monopuce ajoutés à la carte mère d'un ordinateur. D'autres réalisations ajoutent un mode d'exécution spécial au micro-processeurs hôte et y font exécuter le code TPM. L'interaction entre le TPM et le système hôte est très limitée et spécifiée pour garantir que les opérations effectuées dans le TPM et les données associées ne peuvent pas être visualisées de l'extérieur, pas même du logiciel privilégié ou du débogueur (voir [47]). Ces spécifications répondent aux critères de validation FIPS 140 [48].

Le TPM est capable de générer une clé cryptographique unique par instance TPM qui est stockée dans une mémoire non volatile ; en particulier chaque TPM a une clé d'approbation (EK) unique et secrète gravée durant la production du TPM. Le TPM fournit également un générateur de nombres aléatoires qui utilise au moins une source d'entropie telle que le bruit, les variations d'horloge, le mouvement de l'air et d'autres types d'événements.

Les opérations cryptographiques peuvent consommer une grande quantité de ressources de calcul, et plusieurs fournisseurs proposent donc des accélérateurs cryptographiques spécifiques qui peuvent être connectés aux systèmes existants.

Outre les processeurs qui suivent le standard TPM, il existe aussi d'autres types de processeurs sécurisés ne respectant pas la même interface mais offrant des fonctionnalités similaires et appelés « élément sécurisé. » Ceux-ci sont souvent une évolution de la puce traditionnellement utilisée dans les cartes à puce et qui a été adaptée aux besoins de numérisation croissant au sein d'appareils divers[24] (par ex. téléphones, tablettes, voitures, et autres appareils connectés).

3.4 Protection des données

Les secrets sensibles tels que les clés cryptographiques, les chaînes d'authentification ou les données présentant des problèmes de confidentialité peuvent être conservés dans un TEE, et les opérations impliquant ces secrets peuvent être effectuées dans le TEE, éliminant ainsi le besoin d'extraire les secrets en dehors du TEE.

Il existe plusieurs façons de protéger les unes des autres, les applications partageant les mêmes ressources matérielles. Il s'agit notamment des éléments suivants :

- **Chiffrement de la mémoire** : les données stockées dans la mémoire vive partagée sont chiffrées.
- L'**isolation d'application** peut être réalisée à l'aide d'enclaves d'applications distinctes avec des clés

[24] On trouve trois formats pour ces éléments sécurisés : les cartes de circuit intégré universelle (UICC) qui sont essentiellement des cartes à puce sous forme de carte SIM ; les éléments sécurisés embarqués (eSE) qui sont des micro-contrôleurs de cartes à puces intégrés dans des puces de communication en champ proche (NFC) ; et les cartes mémoire sécurisées (microSD).

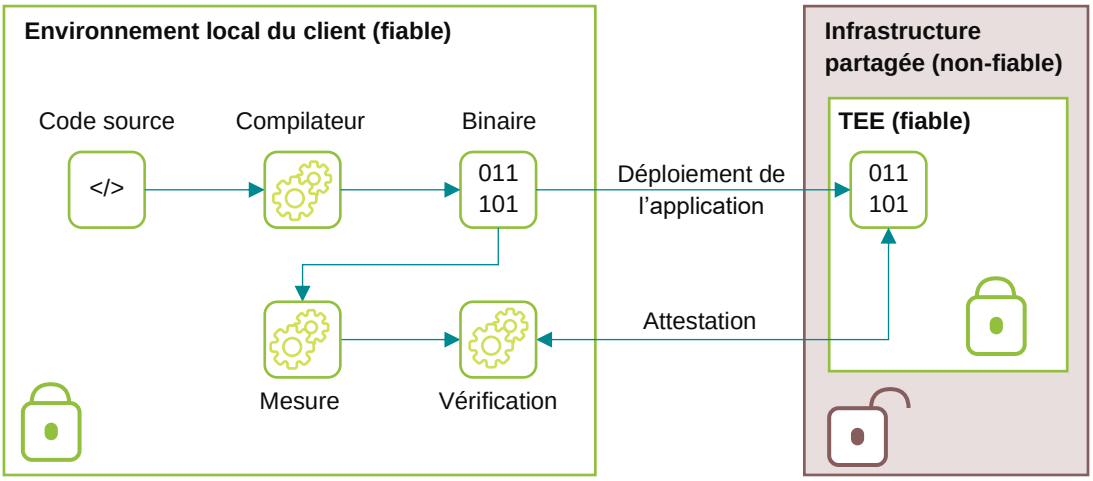


FIGURE 5 – DIAGRAMME GÉNÉRIQUE POUR LE DÉPLOIEMENT ET L'ATTESTATION D'APPLICATIONS DANS UN TEE.

uniques par application. Les développeurs d'applications peuvent intégrer une bibliothèque dans leur application et définir quelles parties du code doivent être exécutées dans le TEE.

- L'**isolation de machine virtuelle** est obtenue en chiffrant la mémoire allouée à la machine virtuelle avec une clé unique et permet de supprimer l'hyperviseur du domaine de confiance.

Un client peut chiffrer des données avec la clé publique du TEE afin que seul le code exécuté à l'intérieur du TEE puisse utiliser ces données. Le client peut vérifier la clé publique à l'aide du mécanisme d'attestation à distance.

3.5 Attestation à distance

L'attestation est un élément fondamental pour vérifier la fiabilité d'un TEE et ensuite décider de lui faire confiance ou pas. Elle permet à un environnement logiciel de garantir qu'un programme spécifique est exécuté par un matériel spécifique. Cette garantie, devrait contenir des informations complètes, récentes et explicites sur le plan sémantique [49]. Une attestation locale peut être utilisée entre deux environnements logiciels partageant le même matériel, mais ce qui intéresse la plupart des clients de systèmes avec TEE, est l'attestation à distance où les deux environnements logiciels fonctionnent sur du matériel différent. Elle permet également l'application de politiques de sécurité spécifiques.

La plupart des TEE peuvent générer des preuves cryptographiques vérifiables, qui peuvent être envoyées au client (la partie évaluatrice), qui peut ensuite valider la signature de la preuve. De manière simplifiée, si la signature et la preuve sont valides, le client conclut que le logiciel attendu est exécuté dans un TEE authentique. La preuve est basée sur la mesure du TEE et peut également inclure une clé publique du TEE (voir 3.2). Comme le confirment des tests effectués sur des infrastructures courantes, il existe cependant une tension inévitable entre ce que le client souhaite recevoir comme information pour prendre sa décision et ce que souhaite ou peut révéler le fournisseur d'infrastructure, tension qui pourrait être désamorcée en effectuant des attestations, peut-être moins détaillées, mais fréquentes [49].

La Figure 5 présente un processus d'attestation générique. La partie utilisatrice du TEE doit connaître la mesure attendue, non seulement de l'application qu'elle souhaite déployer (pouvant inclure le système d'exploitation) mais également du micrologiciel du TEE. La vérification peut s'effectuer grâce à un mécanisme de défi-réponse. Il faut s'assurer que :

- Le **fichier binaire exécuté dans l'enclave a bien été construit avec le code attendu**. Pour ce faire le client peut compiler son application sur une machine de confiance (par ex. lui appartenant), puis copier le binaire de manière sécurisée sur la plateforme d'informatique confidentielle. Une autre méthode est d'utiliser un outil de compilation reproductible[25].

- Le **fichier binaire en cours d'exécution correspond bien au binaire attendu**. Un système d'attestation utilise des clés cryptographiques dérivées de secrets figés dans le microprocesseur de confiance pour signer une preuve que le binaire est dans un état donné sur un véritable matériel (pas

[25] Par exemple Nix [50].

une simulation). La preuve contient une mesure (valeur de hachage cryptographique) du binaire.

- L'**état de l'application au moment de son démarrage** est celui attendu : la mesure de la partie exécutable du binaire n'est pas suffisante pour prédire son comportement futur.
- L'**attestation est signée par une entité de confiance**, en principe par le fabricant du microprocesseur ou de la puce sécurisée.

De cette façon, en supposant que l'on fasse confiance à la sécurité physique du microprocesseur ou de la puce spécialisée, que les attaques connues ont été atténuées et que le code de l'enclave n'est pas vulnérable aux attaques par canaux auxiliaires, l'attestation permet de convaincre le client que l'attestation a bien été produite par un logiciel spécifique à l'intérieur d'un matériel spécifique sans interférence extérieure. Plus de détails ainsi que des variations sont donnés dans [51].

Enfin, le système d'attestation doit permettre de pouvoir révoquer le TEE dans le cas où il s'avère que la sécurité de ce dernier est compromise.

3.6 Efforts de standardisation

Les définitions de TEE diffèrent et sont parfois incomplètes. Cela peut conduire à un faux sens de sécurité, à des incertitudes légales et à rendre les comparaisons difficiles [52]. Il existe deux organisations importantes tentant de standardiser l'informatique confidentielle et les TEE : la « *GlobalPlatform*[26] » (GP) et le « *Confidential Computing Consortium*[27] » (CCC).

GP est une organisation de normalisation technique fondée en 1998. Elle a publié la première spécification

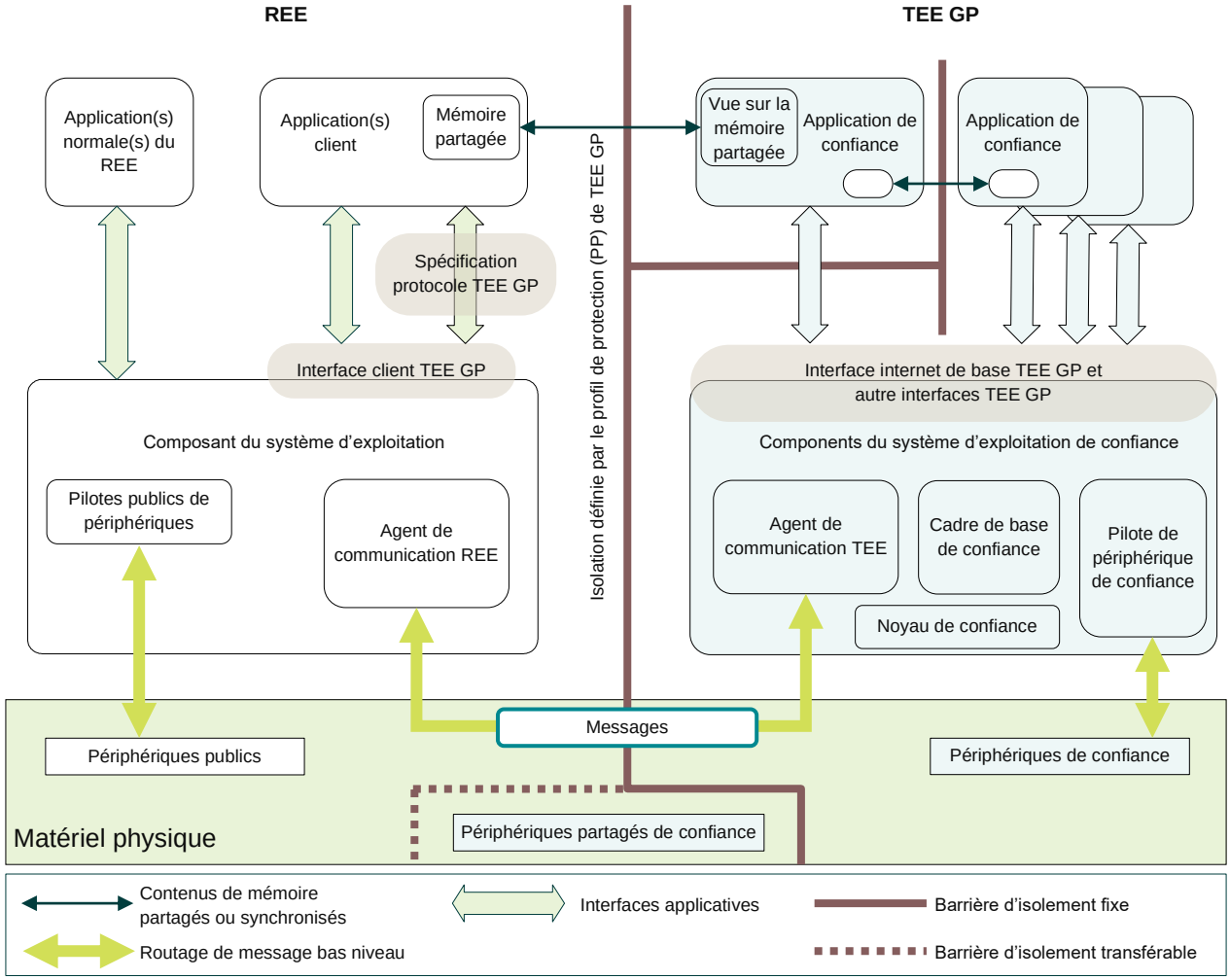


FIGURE 6 – L'ARCHITECTURE LOGICIELLE TEE DE LA GLOBALPLATFORM VISE À PERMETTRE AUX APPLICATIONS DE CONFIANCE DE FOURNIR DES FONCTIONNALITÉS ISOLÉES ET FIABLES QUI PEUVENT ÊTRE UTILISÉES VIA DES APPLICATIONS CLIENTES (D'APRÈS [9]).

[26] https://globalplatform.org/

[27] https://confidentialcomputing.io/

de TEE en 2010. Cette dernière a été fortement inspirée par la proposition TrustZone de la société ARM. La spécification la plus récente concerne une architecture matérielle et logicielle [12] qui met en évidence une isolation complète entre un environnement d'exécution régulier (REE) et le TEE. Dans ces spécifications, chaque environnement a ses propres ressources (par ex., RAM, ROM, CPU, application de confiance, OS, etc.) et les communications entre les deux environnements ne sont effectuées que via l'interface de programmation client du TEE.

La Figure 6 présente les principaux composants logiciels d'un TEE GP. Le noyau de confiance exécute des applications de confiance[28] (TA) et fournit des fonctionnalités aux développeurs. Le noyau de confiance procède à un démarrage sécurisé (authentification et isolement du REE) à chaque allumage. L'interface de programmation interne du TEE est l'interface qui permet aux applications de confiance de communiquer rapidement avec d'autres applications de confiance et d'accéder à des ressources matérielles fiables (via des fonctions approuvées).

La spécification GP fait usage d'un grand nombre de primitives cryptographiques et les met à la disposition de l'utilisateur. Certaines recommandations sont faites concernant leur usage [53] mais le choix final est laissé au fabricant du TEE.

CCC a été créé en 2019 afin de répondre au manque de standardisation et d'interopérabilité entre les différentes approches d'informatique confidentielle. Il promeut l'utilisation généralisée d'environnements d'exécution de confiance, attestés et basés sur le matériel.

3.7 Évaluation des risques

Une attaque réussie sur un composant TEE résulte de la combinaison de plusieurs facteurs dans les phases d'identification (la création d'une attaque jusqu'au moment où elle est exploitable grâce à une description précise, la disponibilité d'outils, etc.) et d'exploitation de l'attaque. Ces facteurs incluent :

- Temps écoulé (heures, semaines, mois, etc.) ;
- Accès à des composants TEE (nombre de composants nécessaires à l'identification et l'exploitation) ;
- Expertise (bonne connaissance, expert du domaine, plusieurs experts) ;
- Connaissance du TEE (informations publiques, restreintes, sensibles ou critiques) ;
- Outils et équipement (standard, professionnel, sur mesure, etc.) ;
- Échantillon sur lequel un attaquant peu installer ou exécuter du code de confiance (public, restreint, etc.).

L'exploitation de l'attaque demande souvent moins d'efforts que son identification. Le rôle principal du TEE est « *d'empêcher des attaques de se répandre largement via Internet (ou d'autres moyens) à de nombreux utilisateurs finaux pour un coût minimal* » [41].

Habituellement, un niveau de risque est estimé sur la base de la probabilité d'un incident mis en correspondance avec l'impact négatif estimé. La probabilité de l'incident est donnée par une menace exploitant une vulnérabilité avec une probabilité donnée.

3.8 Types d'attaques connues

Dans cette section nous listons plusieurs types d'attaques connues pouvant conduire à un vol de données. On peut trouver une liste plus exhaustive dans [41].

Les TEE essayent d'en empêcher la plupart, en particulier les attaques logiques.

3.8.1 Attaques logiques

Les TEE permettent de réduire les problèmes liés aux attaques logiques dues à des vulnérabilités dans l'hyperviseur, le système d'exploitation ou les applications car ils ajoutent des contrôles supplémentaires à l'accès de la mémoire tout en chiffrant son contenu.

En revanche les TEE sont potentiellement eux-mêmes vulnérables à des erreurs dans leur propre code. Plusieurs atténuations sont possibles : réduction de la base de code de confiance, vérification formelle de ce code de confiance, réduction du nombre de fonctions de l'interface de communication avec les applications non sécurisées.

[28] Une application de confiance est une application fonctionnant à l'intérieur d'un TEE.

3.8.1.1 Violation de la sécurité de la mémoire

Les erreurs de mémoire dans les systèmes de bas niveau écrits dans des langages de programmation peu sûrs tels que C ou C++ représentent l'un des principaux problèmes en matière de sécurité informatique [54], [55]. Par exemple les dix principales vulnérabilités du classement MITRE comprennent quatre types d'erreurs de mémoire. Environ 70% des vulnérabilités dans les produits Microsoft corrigés par des mises à jour chaque année sont des problèmes liés à la sécurité de la mémoire [56]. Google signale un nombre similaire de bogues dans son navigateur Chrome.

Il existe plusieurs types de vulnérabilités conduisant à une violation de la sécurité de la mémoire, comme par exemple :

- **Débordement de tampon** – Une vulnérabilité de débordement de tampon se produit lorsque le programmeur d'un logiciel ne vérifie pas les limites sur les opérations de mémoire tampon. Différentes vulnérabilités sont possibles selon l'emplacement de la mémoire (tas[29] ou pile d'exécution[30]).
- **Pointeurs vers objet effacé** – Lors de la destruction d'un objet ou de la désallocation de mémoire, il se peut que la valeur du pointeur ne soit pas modifiée et donc que le pointeur pointe toujours vers l'emplacement de la mémoire désallouée. Si un tel emplacement mémoire est accédé après sa libération, le programme peut lire ou écraser la mémoire de façon arbitraire – au lieu d'émettre une exception si le pointeur avait été mis à NULL.
- **Attaque par traduction d'adresse** – Ces attaques exploitent le mécanisme de traduction entre adresses de mémoire logiques et physiques (les applications fonctionnant sur une même machine physique partagent la même mémoire physique).
- **Mauvaise validation des entrées de l'utilisateur** – Lorsque qu'un logiciel ne vérifie pas les entrées de l'utilisateur, un utilisateur malintentionné peut alors fournir des entrées forgées qui sont inattendues et peuvent entraîner des violations de la sécurité. On compte parmi ces types d'entrées non contrôlées : les pointeurs non contrôlés ou les paramètres hors plage. Ceux-ci peuvent conduire par exemple à une lecture ou une écriture arbitraire en mémoire.

En général la phase d'identification de vulnérabilité conduisant à une violation de la sécurité de la mémoire inclut :

- Tests à données aléatoires afin de trouver des problèmes potentiels ;
- À partir de ces tests, rétro-ingénierie du code binaire afin de déterminer la meilleure façon d'exploiter le problème ;
- Création d'un outil logiciel exploitant la vulnérabilité identifiée.

Des techniques matérielles comme « *capability-based addressing* » permettent de se prémunir des violations de la sécurité de la mémoire mais imposent une réécriture partielle du code afin de les utiliser.

3.8.1.2 Dépassement d'entier

Les vulnérabilités de dépassement d'entier surviennent lorsqu'un programmeur ne prend pas en compte les valeurs minimales et maximales pouvant être stockées dans une variable de type entier. Ces vulnérabilités peuvent être exploitées de différentes manières, par exemple pour contourner certains mécanismes de protection, pour induire des boucles infinies, des erreurs logiques, des élévations de privilèges, etc.

La phase d'identification d'une vulnérabilité par dépassement d'entier est similaire à celle d'une vulnérabilité conduisant à une violation mémoire (voir 3.8.1.1).

3.8.1.3 Hameçonnage d'un appel système

L'hameçonnage d'appels du système est utilisé pour intercepter les appels de fonctions du système afin d'espionner ou de modifier leur comportement. Pour cela il faut en général modifier le système d'exploitation à l'arrêt ou en cours d'exécution.

3.8.1.4 « *Jump-oriented programming* »

« *Jump-oriented programming* » [7] est une nouvelle classe d'attaques exploitant les flux d'exécution des applications, et qui, selon les auteurs est impossible à détecter ou empêcher, à moins d'utiliser un système appliquant une intégrité de flux total (par ex., [57]), au prix d'une complexité accrue d'un impact négatif sur les performances. La phase d'identification implique une rétro-ingénierie du code binaire afin de trouver une vulnérabilité.

[29] Segment de mémoire utilisé lors de l'allocation dynamique de mémoire à la demande d'un programme durant son exécution.

[30] Structure de données servant à enregistrer des informations au sujet des fonctions actives. Sa principale utilisation est de garder la trace de l'endroit où chaque fonction active doit retourner après son exécution.

3.8.1.5 Situation de concurrence

Une hypothèse de concurrence non valide peut conduire à ce genre de vulnérabilité : supposer que l'état de sécurité ne change pas entre le moment où elle est vérifiée et le moment de l'exécution effective de l'opération. Si des changements surviennent effectivement, alors l'opération en question peut conduire à une faille de sécurité moyennant une analyse fine de l'orchestration des instructions. L'identification d'une telle faille requiert de pouvoir tester le système cible.

3.8.1.6 Temps d'exécution

Une attaque temporelle est basée sur la différence de temps dans l'exécution du code. Il faut pouvoir avoir accès à une horloge très précise afin de distinguer différentes opérations (par ex. accès mémoire, le mécanisme d'interruption, changement de branche, etc.). Ici encore, une rétro-ingénierie de l'application cible est requise.

3.8.1.7 Vulnérabilité physique exploitée par le logiciel

Cette classe d'attaques se produit lorsque le logiciel est en mesure d'induire un comportement matériel spécifique qui peut ensuite être exploité par l'attaquant. On trouve ici les injections de fautes (par ex. Rowhammer [58]) ou les attaques de cache (par ex. récupération de clés utilisées par le TEE). Pour ce second type d'attaque plusieurs conditions particulières doivent être réunies et l'identification de la vulnérabilité requiert beaucoup de ressources :

- Accès « *root* » au REE ;
- Documentation très détaillée du TEE ;
- Outils de débogage REE ;
- Possibilité d'exécuter une application de test effectuant des opérations avec la clé ciblée ;
- Possibilité d'effectuer un grand nombre de mesures de l'opération cryptographique TEE au travers du REE.

3.8.2 Attaques requérant un accès physique

Nous présentons ici brièvement quelques attaques parmi les moins sophistiquées sur le matériel. Il en existe d'autres requérant des équipements et une expertise tels que leur coût reste prohibitif. Par exemple certains microscopes permettent d'avoir une image des photons émis par un transistor en fonctionnement [59].

Quoi qu'il en soit la mise en œuvre de ces attaques est d'autant plus difficile dans le cadre d'une infrastructure informatique publique, qu'il faut d'abord avoir accès à la bonne machine dans bâtiment dont les accès sont normalement restreints et très contrôlés.

3.8.2.1 Injection d'erreur

Le but de cette attaque, qui peut être destructive, est d'interférer avec le bon fonctionnement du TEE en utilisant des méthodes physiques (envoi de décharges électriques, d'impulsions laser, d'erreur d'horloge, etc.) et de corrompre le flux d'exécution du code.

Ce type d'attaques requiert :

- Un canal auxiliaire pour identifier une faiblesse du TEE
- L'installation d'outils pour provoquer l'erreur (par ex., générateur de pulse, oscilloscope, etc.)

3.8.2.2 Attaques par canal auxiliaire

Ce type d'attaques, plus connu sous la terminologie anglaise de « *side-channel attack*, » a généralement pour but de récupérer des clés de chiffrement d'un système de chiffrement à travers une mise en œuvre en pleine exécution de ce système[31].

De nombreuses attaques de ce type ont été découvertes sur les cartes à puce et les micro-processeurs [60] et notamment les processeurs d'Intel avec la technique SGX [61]. Elles concernent l'observation minutieuse de caractéristiques comme le temps d'exécution, la consommation d'électricité, les émissions électromagnétiques ou acoustiques, etc. Elles existent en raison de la façon dont les applications (ou librairies tierces) sont écrites.

Ce type d'attaques requiert :

- La possibilité d'effectuer des opérations de chiffrement avec la clé que l'on souhaite récupérer ;
- La possibilité d'effectuer des milliers de mesures, souvent avec des équipements particuliers, d'une caractéristique particulière afin d'établir une statistique ;
- La connaissance approfondie des schémas de chiffrement utilisés.

[31] Ce type d'attaques n'est pas nouveau : les Britanniques en ont fait usage contre l'Ambassade égyptienne à Londres dans les années 1960.

Les attaques par canal auxiliaire ne peuvent être empêchées par le fabricant de matériel seul. De par leur nature elles concernent aussi les développeurs d'applications. Écrire des applications fonctionnant dans des TEE requiert donc un certain niveau d'expertise ainsi qu'une compréhension des méthodes permettant de limiter les attaques par canal auxiliaire. Les fabricants de TEE mettent à disposition des outils de développement spécifiques. Par exemple, Intel propose un nombre de conseils pour les développeurs [62].

3.8.2.3 Sondage externe de mémoire vive dynamique

Le but de cette attaque est de sonder des données sur un bus de mémoire vive externe pour trouver un secret du TEE ou d'une application de confiance. Pour cela il faut que la mémoire vive ne soit pas chiffrée et que le texte clair soit connu.

L'une des difficultés de ces attaques est que les schémas d'accès et les adresses mémoires observés sur la mémoire physique sont très différents de ceux de l'application en raison des hiérarchies de cache complexes des processeurs moderne. Cela requiert une rétro-ingénierie des règles de traduction mémoire internes au micro-processeurs [63] ainsi que :

- La connaissance de la fréquence du bus de communication utilisé entre la mémoire vive et le TEE ;
- Un outil d'analyse permettant d'enregistrer les communications sur le bus ;
- L'analyse cryptographique des données recueillies pour récupérer la clé.

L'utilisation de mémoire « inconsciente » ([64], [65]) qui permet de brouiller les schémas d'accès à la mémoire vive d'un programme, est une défense possible, mais son coût en terme de performance rend la technique prohibitive pour le moment. Une alternative est de chiffrer le bus de communication, mais cela introduirait des coûts supplémentaires importants aux barrettes de mémoire. D'autres alternatives existent mais introduisent un coût significatif et ne sont pas déployées.

3.8.2.4 Interface de débogage non-protégée

Le but de cette attaque est d'accéder, de lire ou de modifier directement le contenu de la mémoire du TEE au moyen d'une fonction de débogage matériel, et d'utiliser ce privilège pour trouver une vulnérabilité exploitable par un logiciel, par exemple en modifiant des registres permettant une exécution avec privilège sans autorisation.

Il faut cependant être capable d' :

- Analyser le système auquel les broches auxiliaires d'entrée-sortie des composants intégrés donnent accès ;
- Analyser le logiciel du TEE et d'y trouver une vulnérabilité exploitable ou de modifier le contenu de la mémoire du TEE afin de modifier des valeurs d'autorisation.

3.9 Mise en œuvre

Dans cette section nous regardons plus en détail le fonctionnement des principales mises en œuvre commerciales de TEE. Comme nous le signalions dans la section 3.6, les définitions d'informatique confidentielle et de TEE diffèrent et sont parfois incomplètes. En particulier, notons que l'approche utilisée par AWS pour son système Nitro (voir 3.9.2.1) est très différente des autres[32].

3.9.1 Fabricants de microprocesseurs

AMD et Intel sont les deux principaux fabricants de microprocesseurs offrant des fonctionnalités nécessaires à l'informatique confidentielle dans de larges centres informatiques[33]. Alors que les techniques TDX d'Intel et SEV-SNP d'AMD ont pour but de protéger des machines virtuelles (VM) entières, la technique SGX est très différente et sa surface d'exposition aux attaques plus faible. La Figure 7 montre les éléments faisant partie de la base informatique de confiance pour ces trois techniques que nous décrivons dans les paragraphes suivants.

[32] Par exemple le terme « enclave » n'a pas du tout le même sens chez Amazon et Intel. Alors que les « Nitro Enclaves » sont des machines virtuelles entières, les « SGX Enclaves » sont des bibliothèques exposant certaines API.

[33] Notons que NVIDIA offre depuis peu un processeur graphique (A100 Tensor Core avec la technique « *Ampere Protected Memory (APM)* ») qui introduit un mode d'exécution confidentielle dans le processeur graphique et permet ainsi d'utiliser des ensembles des données pour former et déployer des modèles d'apprentissage automatique (« *machine learning* ») de manière confidentielle, notamment sur une infrastructure informatique publique. Ces processeurs sont disponibles sur l'offre Azure de Microsoft.

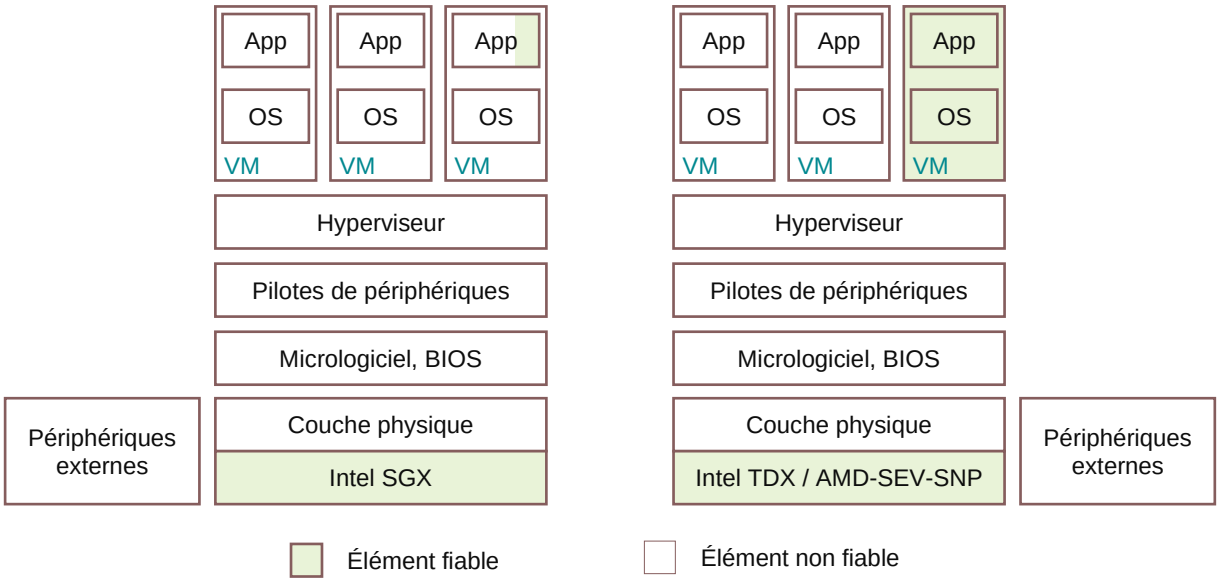


FIGURE 7 – LA TECHNIQUE SGX D'INTEL PERMET D'ISOLER UN PROCESSUS, TANDIS QUE LES TECHNIQUES TDX D'INTEL ET SEV-SNP D'AMD PERMETTENT L'ISOLATION DE MACHINES VIRTUELLES ENTIÈRES.

3.9.1.1 Technique SEV-SNP d'AMD

En 2016, AMD a introduit la technique de virtualisation sécurisée par chiffrement (« *Secure Encrypted Virtualisation (SEV)* ») afin d'isoler les VM de l'hyperviseur au niveau matériel [66], [67]. Chaque VM reçoit sa propre clé de chiffrement AES pour le chiffrement de la mémoire. L'état des registres du microprocesseur de chaque VM est également chiffré, empêchant l'hyperviseur de lire les données contenues dans la VM. Par la suite AMD a ajouté une technique de pagination imbriquée sécurisée (« *Secure Nested Paging (SNP)* ») offrant une protection de l'intégrité de la mémoire, et permettant d'empêcher les attaques d'un hyperviseur malicieux (par ex. attaques par rejeu, reconfiguration du mécanisme de traduction de mémoire virtuelle, corruption des données mémoires) [68]. Le principe fondamental de SEV-SNP est que si une VM est en mesure de lire une page mémoire lui étant réservée (et donc chiffrée), alors elle doit toujours lire la dernière valeur qu'elle a elle-même écrite. Par ailleurs dans le modèle de sécurité utilisé pour SNP, seule la VM du client et le microprocesseur AMD font partie de la base informatique de confiance. N'en font donc pas partie l'hyperviseur, le BIOS, les autres VM, etc. (voir Figure 7). Enfin, une option de SEV-SNP permet aux VM de diviser, d'une manière similaire aux anneaux de protection dans l'architecture x86, leur mémoire virtuelle en quatre niveaux de privilèges (« *Virtual Machine Privilege Levels (VMPL)* »).

Le mécanisme d'attestation à distance d'AMD, permet de vérifier que la machine hôte est bien un processeur AMD qui prend en charge la technique SEV-SNP et qu'une VM a bien été déployée avec la protection SEV-SNP. Chaque processeur AMD, contient un coprocesseur sécurisé qui permet de générer une paire de clés dédiées (« *Platform Endorsement Key (PEK)* »), elle-même signée par une clé unique dérivée de secrets enregistrés grâce à des fusibles dans la puce elle-même. Cette PEK est également indirectement utilisée pour établir un secret partagé entre la plate-forme SEV et le client [69]. Au moment du lancement par l'hyperviseur de la VM sécurisée sur la plate-forme SEV, le micrologiciel SEV calcule la mesure (valeur du hachage cryptographique) de la mémoire de la VM. Cette mesure peut être communiquée de manière sécurisée au client afin qu'il vérifie que la VM déployée n'a pas été altérée.

La technique SEV-SNP est disponible sur les processeurs AMD EPYC de 3ème génération (série 7003[34]) et 4ème génération (série 9004[35]). On peut trouver ces processeurs chez différents fournisseurs comme Dell (serveurs « PowerEdge[36] »), Lenovo[37] ou HP[38]. Les prix varient de quelques milliers à plusieurs dizaines de milliers d'euros en fonction de la configuration.

34 https://www.amd.com/fr/processors/epyc-7003-series
35 https://www.amd.com/fr/processors/epyc-9004-series
36 https://www.dell.com/fr-be/shop/serveurs-dell-poweredge/sr/servers/amd?appliedRefinements=34947
37 https://www.lenovo.com/be/fr/data-center/alliance/amd/
38 https://www.hpe.com/us/en/solutions/amd.html

3.9.1.2 Techniques SGX et TDX d'Intel

Introduit en 2015, le système « *Software Guard eXtensions (SGX)* » d'Intel permet à un logiciel de définir des zones de mémoire protégées pour des « enclaves » sécurisées isolées des autres processus fonctionnant sur la même machine (noyaux du système d'exploitation, hyperviseur, etc.) ainsi que des accès directs par des périphériques. Le processeur s'assure que chaque enclave a sa zone mémoire dédiée et chiffrée, enregistrant chaque allocation par le système d'exploitation [70]. Une enclave est générée en tant que bibliothèque partagée dynamiquement à l'aide d'outils de compilation standards. Lors de l'initialisation d'une enclave, le système d'exploitation demande au processeur de copier l'application dans des pages mémoire de la zone protégée chiffrée. Lors de ce chargement en mémoire le processeur calcule une mesure de l'application. Cela permet par la suite de vérifier l'intégrité de l'application par un mécanisme d'attestation. Intel a introduit en 2020, la technique « *Trusted Domain Extensions (TDX)*, » qui est un module logiciel signé, exécuté dans un nouveau mode du processeur et qui permet de protéger et d'isoler cryptographiquement des machines virtuelles. Plus de détails sur le fonctionnement et l'architecture de TDX peuvent être trouvés dans [71].

Deux types d'attestation à distance sont disponibles avec SGX « *Enhanced Privacy ID (EPID)* » et « *Data Centre Attestation Primitives (DCAP)*. » Le premier est un mode d'attestation dans lequel le serveur d'attestation d'Intel doit être contacté pour obtenir des informations sur l'enclave requérante. Le second ne nécessite pas de contacter le serveur d'attestation d'Intel. Durant le processus de construction d'une enclave, deux mesures sont faites. `MRENCLAVE` est la valeur de hachage cryptographique de la disposition de la mémoire virtuelle assignée à l'enclave au moment de son lancement. L'autre mesure, `MRSIGNER`, est la valeur de hachage cryptographique de la clé publique de l'auteur de l'application fonctionnant dans l'enclave [72].

Alors que la technique SGX a été retirée de la 12ème génération des processeurs Core d'Intel, elle reste disponible sur la 3ème génération de processeurs Xeon [73]. Les processeurs Xeon de 4ème génération prennent en charge la technique TDX et sont disponibles chez les partenaires et distributeurs agréés d'Intel[39].

Notons que l'utilisation de la technique SGX demande une réécriture importante des applications existantes[40]. Il est en effet nécessaire de partitionner l'application en identifiant quelle partie du code doit pouvoir avoir accès aux données sensibles. Bien que cette étape essentielle soit complexe, elle permet en principe d'améliorer la sécurité de l'application étant donné qu'il est généralement accepté qu'une application de petite taille – en l'occurrence celle tournant dans l'enclave – a moins de chances d'avoir des défauts tout en étant plus facilement vérifiable qu'une application de grande taille. La communication entre la partie sécurisée de l'application dans l'enclave et le reste de l'application se fait via des appels à des fonctions devant être déclarées avant le lancement de l'enclave. Enfin, les applications écrites pour la plate-forme SGX ne peuvent pas être utilisées sur d'autres plates-formes.

3.9.2 Fournisseurs d'infrastructures informatique

Plusieurs fournisseurs d'infrastructures informatiques publiques proposent aujourd'hui des solutions d'informatique confidentielle basées sur les TEE. Nous décrivons ici les trois principaux[41].

3.9.2.1 AWS

Amazon définit l'informatique confidentielle comme l'utilisation de matériel spécialisé et de micrologiciels (« *firmware* ») associés pour protéger le code et les données du client pendant le traitement contre tout accès extérieur. Amazon traduit cela selon deux dimensions :

- la protection vis-à-vis de de l'opérateur de l'infrastructure informatique sous-jacente, en l'occurrence AWS.
- la capacité des clients à diviser leurs propres charges de travail en composants plus ou moins fiables, ou à concevoir des systèmes multi-agents.

L'accent est mis par Amazon sur l'architecture du système Nitro, plutôt que sur la disponibilité d'un micro-processeur particulier en charge de fournir un TEE. Cependant, depuis avril 2023, AWS offre aussi la possibilité de créer des instances EC2 avec la technique SEV-SNP d'AMD (voir 3.9.1.1).

[39] https://www.intel.com/content/www/us/en/partner/showcase/partner-directory/overview.html

[40] Ce n'est pas le cas avec TDX qui permet de protéger une machine virtuelle entière.

[41] Selon le Financial Times, les trois fournisseurs d'infrastructure informatique les plus importants en 2021 étaient Amazon, Microsoft et Google [74].

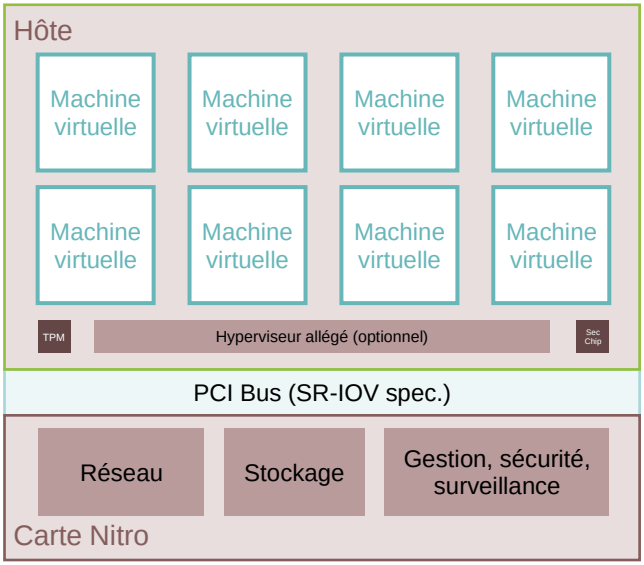


FIGURE 8 – ARCHITECTURE D'UNE MACHINE AWS.

Dans sa conception et selon AWS, le système Nitro n'offre aucun mécanisme permettant à un système ou à une personne de se connecter aux serveurs EC2, de lire la mémoire des instances EC2 ou d'accéder aux données stockées. Les travaux de maintenance ne peuvent se faire qu'à travers des API limitées.

Le système AWS Nitro (Figure 8) s'inscrit dans une refonte de l'infrastructure de virtualisation du service EC2 d'Amazon, réduisant notamment au maximum les parties de l'hyperviseur fonctionnant dans le microprocesseur principal de la carte mère. Le système AWS Nitro est une combinaison de serveurs, de processeurs, de composants de gestion et de micrologiciels spécialisés qui fournissent la plate-forme sous-jacente pour toutes les instances Amazon EC2. Il se compose de trois éléments principaux :

- **Cartes Nitro spécifiques** – Dispositifs matériels conçus par AWS qui assurent le contrôle global du système et la virtualisation des entrées/sorties indépendamment de la carte mère du système avec ses processeurs et sa mémoire.
- **La puce de sécurité Nitro** – Celle-ci est intégrée à la carte mère du serveur et doit permettre un démarrage sécurisé basé sur une racine de confiance matérielle, la capacité d'offrir des instances « *bare metal* » (permettant de se passer de l'hyperviseur d'AWS), ainsi qu'une protection du serveur contre les modifications non autorisées du micrologiciel du système.

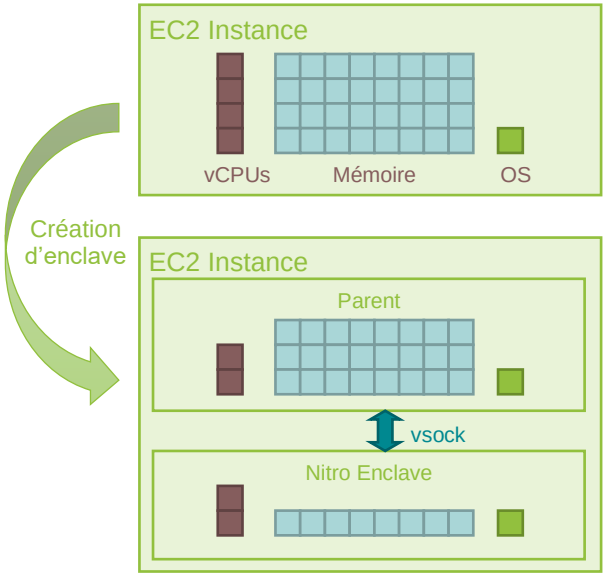


FIGURE 9 – CRÉATION D'UNE ENCLAVE NITRO À PARTIR D'UNE INSTANCE EC2. UNE ENCLAVE EST CRÉÉE EN PARTITIONNANT LE PROCESSEUR ET LA MÉMOIRE D'UNE INSTANCE EC2, APPELÉE INSTANCE PARENT. IL EST POSSIBLE DE CRÉER DES ENCLAVES AVEC DES COMBINAISONS VARIÉES DE CŒURS DE PROCESSEUR ET DE MÉMOIRE.

- **L'hyperviseur Nitro** – Un hyperviseur minimisé, semblable à un micrologiciel, conçu pour fournir une isolation des ressources et des performances.

Les considérations de sécurité de ce système sont détaillées dans [75].

Les enclaves Nitro quant à elles, sont des machines virtuelles isolées fonctionnant avec une instance EC2 classique, appelée instance « parent » (Figure 9). Selon AWS, l'enclave Nitro n'offre pas de sécurité supplémentaire vis-à-vis d'une opératrice d'AWS [76], mais permet en revanche d'empêcher un administrateur du client d'accéder au contenu de l'enclave (code et données).

La contrainte principale imposée par les enclaves Nitro est que l'application fonctionnant dans l'enclave n'a pas de connexion au réseau. Elle peut seulement communiquer avec l'instance parent via une interface point à point appelée « vsock » qui est définie par un identifiant de contexte et un numéro de port[42]. Un simple « *lift and shift* » n'est donc pas possible.

Au moment de sa création, l'application fonctionnant dans l'enclave peut générer une paire de clés asymétriques et faire inclure la clé publique dans l'attestation. Par conséquent l'application client vérifiant l'attestation

[42] Cette « vsock » utilise la même API que les « sockets » POSIX.

peut utiliser cette clé afin d'établir une communication sécurisée avec l'enclave.

3.9.2.2 Microsoft Azure

Microsoft Azure propose trois types d'informatique confidentielle basée sur les TEE :

- Les **enclaves d'applications** sont basées sur la technique SGX d'Intel. Comme indiqué dans la section 3.9.1.2, il est nécessaire de modifier en profondeur les applications existantes afin de les adapter. Cela impose une réflexion importante sur le choix des parties de l'application à sécuriser et de leur interaction avec les autres parties, mais l'avantage de cette approche est de réduire la quantité de code auquel il faut faire confiance. L'inconvénient est bien évidemment la complexité de la mise en œuvre, requérant notamment une formation particulière des analystes, architectes et programmeurs.
- Les **machines virtuelles confidentielles** utilisent la technique SEV-SNP d'AMD (voir 3.9.1.1) et Microsoft a annoncé en avril 2023, la mise à disposition prochaine de la technique TDX d'Intel (voir 3.9.1.2) [77]. Azure met également à disposition un module de plate-forme de confiance utilisé notamment pour l'attestation des machines virtuelles.
- Les **conteneurs confidentiels** permettent au client d'avoir un niveau de contrôle plus fin que les machines virtuelles sur la base informatique de confiance (TCB). Ce modèle d'empaquetage permet en principe d'exécuter des conteneurs existants dans une enclave SGX sans devoir modifier ou recompiler le logiciel (« *lift-and-shift* »).

L'option permettant l'existence de plusieurs fils d'exécution en parallèle (technique « *hyper-threading* ») au sein du même processeur est désactivée sur toutes les instances SGX. Cela permet d'éviter des attaques conduisant à des fuites de données entre applications partageant le même processeur.

3.9.2.3 Google

Google propose différentes façons de mettre en œuvre l'informatique confidentielle sur son infrastructure :

- Les **machines virtuelles confidentielles** utilisent la technique d'AMD.
- Les **nœuds Kubernetes confidentiels** reposent également sur la technique SEV d'AMD. Le moteur Kubernetes de Google peut imposer l'utilisation de machines virtuelles confidentielles pour tous les nœuds Kubernetes.

Alors que toutes les machines virtuelles confidentielles contiennent des TPM virtuels qui valident l'intégrité d'une machine virtuelle avec le démarrage mesuré, les machines virtuelles confidentielles avec la technique SEV-SNP offrent également des rapports d'attestation signés cryptographiquement par le matériel. Cependant la technique SEV-SNP n'est pas encore disponible de manière généralisée sur l'infrastructure de Google.

3.10 Conclusion

Issus du monde des télécommunications[43], les TEE étaient d'abord envisagés dans le cadre de la protection des appareils électroniques grand public intégrés, et notamment les téléphones et décodeurs de télévision. Mais le concept s'est rapidement étendu à l'informatique générale et les fabricants de micro-processeurs ont emboîté le pas, la demande croissante de confidentialité dans les infrastructures informatiques publiques donnant un nouveau souffle à la technique.

Les offres d'informatiques confidentielles basées sur des TEE varient entre les fournisseurs, notamment en fonction du type d'abstraction offert : librairie logicielle, conteneur, machine virtuelle. Le choix de ces options conduit à des différences dans la taille du code de la base informatique de confiance (et donc de la surface exposée aux attaques), mais également dans l'effort d'adaptation nécessaire des applications existantes à ces nouveaux environnements.

Lorsqu'ils sont mis en œuvre correctement, et toutes autres choses égales par elles-mêmes, les TEE fondés sur des composants physiques, permettent d'augmenter significativement le niveau de protection des données au sein d'une infrastructure informatique, en particulier vis-à-vis de tiers, et notamment de cybercriminels[44]. En effet ils permettent d'éviter la plupart des attaques logiques affectant les systèmes habituels, et ce, grâce à une meilleure isolation des processus, un chiffrement de la mémoire par la couche matérielle, un démarrage sécurisé, des mécanismes de contrôle de mise à jour du micrologiciel, et, d'une manière générale à une réduction de la taille de la base informatique de confiance. En d'autres termes les mesures de sécurité

[43] L'une des premières références aux TEE se trouve dans un rapport technique initialement confidentiel de l'« *Open Mobile Terminal Platform* » (OMTP) [78].

[44] « Les données de santé sont attrayantes pour les cybercriminels car elles contiennent des données financières et personnelles, peuvent être utilisées pour le chantage et, surtout, sont idéales pour la facturation frauduleuse » [79].

mises en place dans les TEE rendent une attaque beaucoup plus complexe et coûteuse.

Ces environnements bénéficiant d'importants efforts de standardisation offrent donc une base solide pour le traitement pratique et évolutif de données relativement sensibles dans des centres informatiques gérés par des tiers.

Néanmoins les techniques d'attestation qui devraient permettre d'empêcher la simulation du matériel et de vérifier l'intégrité des applications exécutées restent encore limitées lorsque l'on considère le niveau de confiance qu'il faut accorder au fournisseur d'infrastructure. En effet, dans les solutions étudiées, à l'exception des enclaves SGX, les expériences que nous avons réalisées montrent que soit l'attestation est signée par le fournisseur de l'infrastructure (et non par le fabricant du matériel), soit il n'est pas possible de vérifier la mesure du logiciel sécurisé car il inclut des bibliothèques propriétaires. Le risque est donc que l'entité attestée mente sur son état.

Lors du choix d'une solution d'informatique confidentielle basée sur les TEE, il conviendra de vérifier au minimum les points suivants :

- La protection doit être ancrée dans la couche physique du système et chaque appareil doit avoir une identité unique ;
- Le mécanisme d'attestation doit permettre de pouvoir vérifier le contenu[45] du TEE de manière indépendante du fournisseur d'infrastructure[46] ;
- L'attestation doit être signée au moins par le fabricant du composant physique et pas uniquement par le fournisseur d'infrastructure[47] ;
- Le service permet au minimum d'importer ses propres clés cryptographiques dans un HSM dédié, et au mieux d'utiliser son propre HSM ;
- Il devrait être possible de vérifier le code source des librairies critiques incluses par le fournisseur d'infrastructure dans la base informatique de confiance pour le bon fonctionnement de l'application du client ;
- Un service permettant de traquer les différentes dépendances logicielles, d'environnement de compilation, et des binaires utilisés pour le TEE devrait être mis à disposition du client par le fournisseur d'infrastructure.

[45] Ce contenu, dans le cas de machines virtuelles confidentielles, inclut également le système d'exploitation que l'on préférera de taille minimale.

[46] Si le fournisseur d'infrastructure contrôle en partie le contenu de la machine virtuelle confidentielle ou du conteneur confidentiel et que le client n'a pas de mécanisme pour le vérifier, alors la confiance dans le fournisseur reste totale.

[47] Si le fournisseur d'infrastructure signe l'attestation, alors la confiance dans ce fournisseur reste totale.

4 Comparaisons

Cette section tente de comparer selon différents critères les trois grands types d'informatique confidentielle décrits dans les sections précédentes.

TABLEAU 1 – COMPARAISON SELON PLUSIEURS CRITÈRES DES TROIS PRINCIPALES TECHNIQUES D'INFORMATIQUE CONFIDENTIELLE. LÉGENDE : ✓ : REMPLIT LES ATTENTES ; × : NE REMPLIT PAS LES ATTENTES ; △ : REMPLIT PARTIELLEMENT LES ATTENTES. SOURCES : [16], [80]–[82].

Critère	HE	MPC	TEE
Aucune interaction requise	✓ Oui	× Non[48]	✓ Oui
Aucun matériel spécialisé requis (par ex., processeur)	✓ Oui	✓ Oui	× Non
Domaine de confiance unique	✓ Oui	× Non	✓ Oui
Surcharge de stockage	× Élevé[49]	× Élevé	✓ Quasi nulle
Performance sur calculs simples[50]	△ Faible-Correct	△ Correct	✓ Haute, proche du processeur sans TEE (20 à 2x)
Performance sur calculs complexes[51]	× Très faible	× Faible	✓ Haute, proche du processeur sans TEE (20 à 2x)
Attaques par canaux auxiliaires	✓ Non	× Vulnérable (chez les parties honnêtes)	× Vulnérable
Vérifiabilité des calculs	△ Dépend de la méthode et du calcul	△ Dépend du protocole utilisé	✓ Oui avec le système d'attestation
Logiciel libre	✓ Oui	✓ Oui	△ Dépend du fabricant
Intégrité des données	△ Non	△ Clés seulement	✓ Oui
Confidentialité des données	✓ Oui	△ Clés seulement	✓ Oui
Confidentialité du code	× Non	✓ Oui	✓ Oui
Démarrage authentifié	× Non	× Non	△ Varie

[48] Étant donné que MPC est un protocole distribué, des données sont transmises sur le réseau entre les participants. En particulier les protocoles MPC basés sur des circuits brouillés ont tendance à avoir une complexité de communication élevée (beaucoup de bits). Les protocoles MPC basés sur le partage de secrets ont, eux, tendance à avoir beaucoup d'interactions (beaucoup d'allers et retours).

[49] Par exemple la librairie de chiffrement homomorphe de Google convertit chaque bit de donnée en une liste d'environs 700 entiers de 32 bits chacun, une explosion de la taille des données par un facteur d'environ 22,000 [83] !

[50] Par exemple : sommes, dénombrements, calculs de polynômes de degré faible.

[51] Par exemple : analyse de bases de données, entrainement d'un réseau de neurones pour l'apprentissage automatisé.

TABLEAU 2 – FORCES ET FAIBLESSES DES TROIS PRINCIPALES TECHNIQUES D'INFORMATIQUE CONFIDENTIELLE.

Qualité	HE	MPC	TEE
Forces	• Sécurité grâce à la cryptographie. • Plus sécurisé et général. • Pas besoin de matériel spécialisé. • Certains schémas robustes à des attaques post-quantiques.	• Sécurité grâce à la cryptographie[52]. • Permet le calcul entre plusieurs agents sans tiers de confiance. • Pas besoin de matériel spécialisé (par ex. microprocesseur).	• Sécurité grâce au matériel. • Disponible chez les principaux fournisseurs d'infrastructures informatiques. • Lorsque l'ensemble d'une machine virtuelle est protégée, les applications peuvent être migrées sans ou avec peu de modifications [85].
Faiblesses	• Assez inefficace. • Pas de protection contre les attaquants malintentionnés (à moins d'utiliser des preuves à divulgation nulle de connaissance). • Importante surcharge de calcul. • Malléable[53] par construction et donc sensible à une manipulation du code. • Manque de flexibilité lorsque plusieurs agents souhaitent collaborer. • Récriture nécessaire des applications.	• Protocole complexe difficile à vérifier formellement[54]. • Exigence d'être connecté et fardeau de stockage pour les clients. • Déploiement à travers plusieurs domaines de confiance. • Important coût de communication. • Complexité de la mise en place et de la gestion. • Récriture nécessaire des applications.	• Vulnérable à certaines attaques physiques et logicielles. • Besoin de matériel spécialisé. • Les abstractions de différents fournisseurs peuvent différer et conduire à une dépendance à un fournisseur particulier [85]. • Les systèmes d'attestation ne sont pas publics et donc difficiles à contrôler [85].

[52] Les protocoles MPC sont complexes, et leur étude formelle, afin d'étudier leur sécurité, n'est pas triviale [84].
[53] Un protocole cryptographique est dit malléable si, pour une fonction f donnée, il est possible de transformer le chiffré d'un message m en le chiffré du message $f(m)$ sans connaître m.
[54] Ibid., 52.

Dans un rapport des Nations Unies sur les outils de protection de la vie privée [82], plusieurs techniques d'informatique confidentielle sont comparées selon différents critères. Il ressort que les TEE sont considérés comme les plus matures, devant les méthodes de calcul multipartite, et le chiffrement homomorphe (voir [82]-Figure 6).

La Figure 10 reprend dans le même diagramme de Wardley, les données de ce rapport. L'estimation de la maturité des trois techniques décrites utilise l'échelle suivante :

- Nouveauté : technique rare, peu connue et incertaine. Elle a le potentiel d'avoir une valeur future élevée.
- Émergeant : de plus en plus de personnes commencent à utiliser et comprendre la technique.
- Bon : l'utilisation augmente rapidement à mesure que le marché se développe. La technique est rentable et de nouvelles fonctionnalités sont apportées.
- Très bon : la technique est répandue et stabilisée. L'efficacité opérationnelle est reine et l'échec n'est pas toléré sur le marché.

Selon ce rapport des Nations Unies, la théorie de fonctionnement des TEE est à un niveau de maturité technique relativement élevé. Cependant, une grande partie de ce qu'un utilisateur final attend en termes de convivialité d'un produit informatique est encore relativement nouveau dans le développement des TEE. Cela dit, il existe des produits et services émergents qui prennent en charge les TEE. Certaines infrastructures informatiques, telles que celles de Microsoft et IBM, offrent une fonctionnalité TEE, tandis que d'autres, tels que les services d'Amazon, offrent une technique alternative. La principale lacune à l'heure actuelle est le manque d'environnements de développement faciles à utiliser pour les TEE, ce qui permettrait aux programmeurs généraux d'utiliser ces capacités efficacement et de les configurer correctement.

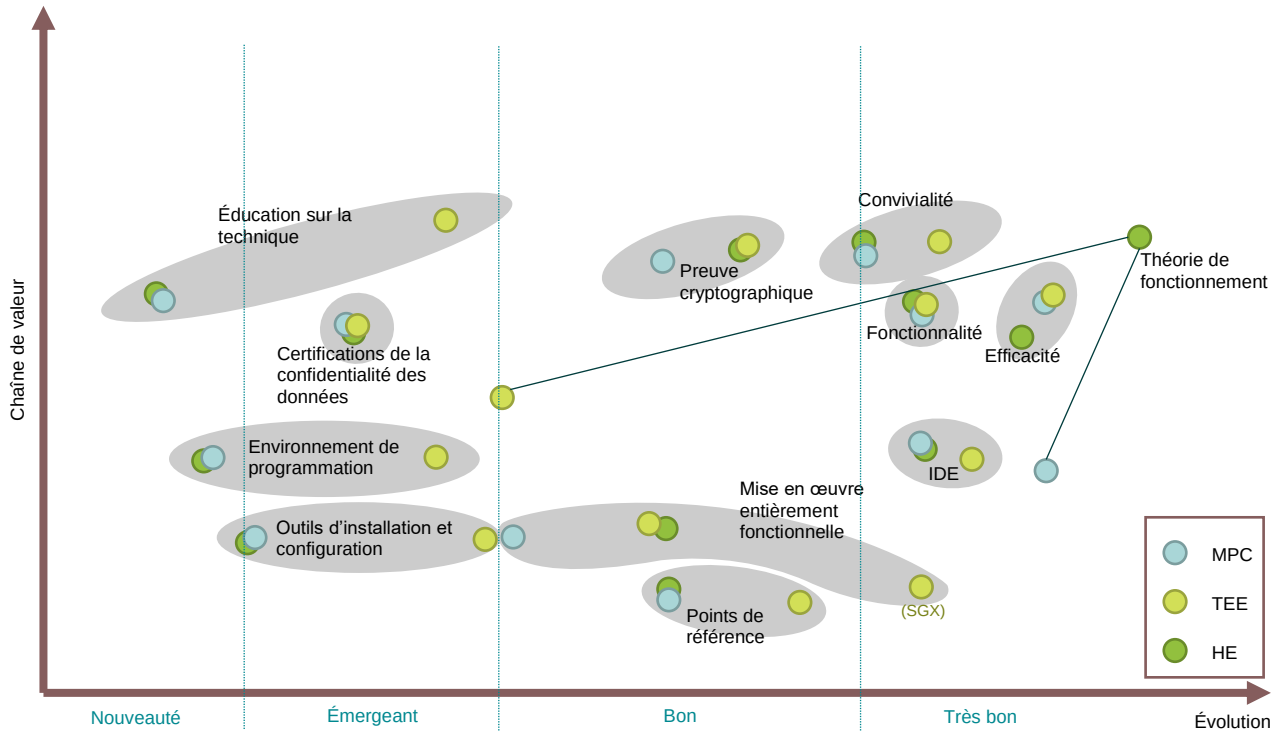


FIGURE 10 – POSITIONNEMENT SUR UN DIAGRAMME DE WARDLEY DES TROIS TECHNIQUES D'INFORMATIQUE CONFIDENTIELLE DÉCRITES DANS CE RAPPORT, SELON L'ÉVALUATION DE [66]. LES ZONES BLEUES ET LES LIGNES NOIRES REGROUPENT VISUELLEMENT LES POINTS CORRESPONDANT AU MÊME CRITÈRE. ON REMARQUE QUE SELON TOUS LES CRITÈRES, SAUF CELUI DE THÉORIE DU FONCTIONNEMENT, LES TEE SONT PLUS AVANCÉS QUE LE HE ET LE MPC.

5 Applications possibles

L'informatique confidentielle permet à un agent de confier ses calculs à un autre agent et introduit même de nouveaux schémas de calculs, comme avec le calcul multipartite. Nous donnons ici deux exemples d'applications possibles de l'informatique confidentielle.

5.1 Études statistiques et apprentissage automatique

Les études statistiques et l'apprentissage automatique (« *machine learning (ML)* ») dans le secteur de la santé et de la sécurité sociale nécessitent souvent des ressources importantes de calcul et de stockage. L'utilisation d'infrastructures informatiques publiques, peut apparaître comme une réponse à ces besoins qui peuvent aussi être variables.

Cependant la nature des informations traitées soulève des questions concernant la préservation de la vie privée et de la confidentialité des données plus ou moins sensibles à traiter. L'utilisation des TEE permet de réduire les risques de fuite de données tout en maintenant un bon niveau de performance.

Dans le cas de l'apprentissage automatique, le besoin de protection s'applique non seulement aux données utilisées pour la phase d'apprentissage, mais également au modèle calculé ainsi qu'aux données d'entrée du modèle en cours d'utilisation. Les TEE permettent d'offrir une protection supplémentaire dans ces trois cas. On peut alors imaginer un scénario où :

- L'agent A qui dispose de données sensibles, n'a pas la puissance informatique suffisante pour construire un modèle de ML. Il utilise donc une infrastructure publique : les données chiffrées sont envoyées au moteur d'apprentissage qui fonctionne sur cette infrastructure dans un TEE. Le fournisseur d'infrastructure ne peut donc accéder aux données.
- L'agent A confie à l'agent B le modèle chiffré ainsi que la mise en œuvre d'un service en ligne fonctionnant sur une infrastructure publique.
- Le TEE fonctionnant sur l'infrastructure informatique publique applique une politique d'exécution qui permet de déchiffrer le modèle et les données envoyées par des utilisateurs finaux via le service en ligne au sein d'une enclave de telle façon que ni l'agent B, ni le fournisseur d'infrastructure n'ont accès au modèle ou aux données.

5.2 Renforcement de la sécurité d'une infrastructure

L'utilisation d'infrastructures informatiques partagées est source de flexibilité d'efficacité et d'innovation mais soulève des questions concernant la souveraineté des données. Certains États ont donc choisi de créer leurs propres infrastructures informatiques pour leurs données et leurs services. Celles-ci peuvent alors être partagées entre différents services de l'État conduisant en outre à des réductions non négligeables de coûts[55].

Les exigences en termes de sécurité de ces infrastructures sont importantes et complexes :

- D'abord leur nature suscite l'intérêt de cybercriminels ou d'agences de renseignement d'autres États.
- Ensuite, afin de gérer ce type d'infrastructure l'État peut faire appel à des sociétés privées nationales ou internationales, par définition en dehors de son contrôle.
- De plus, la mise en œuvre de certains services peut être confiée à différents prestataires, eux aussi en dehors du contrôle de l'État.
- Enfin, différents services de l'État peuvent également requérir des niveaux de sécurité différents : on imagine en effet facilement que les exigences du

[55] Il était estimé en 2020, que l'infrastructure G-Cloud permettrait d'économiser environ 37 M€ grâce à des synergies intelligentes [86].

service des statistiques officielles, ne sont pas les mêmes que celles des services de santé ou encore celles de la défense.

Comme nous l'avons vu précédemment, le matériel spécialisé des TEE renforce le cloisonnement entre applications ou machines virtuelles, et, par conséquent permet de réduire l'impact d'une attaque extérieure qui pourrait compromettre un service particulier. De plus le mécanisme d'attestation d'un TEE peut être utilisé dans le cadre d'une séparation des tâches : l'agent faisant appel au TEE ne doit pas nécessairement faire confiance au gestionnaire de l'infrastructure pour l'exécution de son code.

Par conséquent, l'ajout de TEE, en complément des mesures et politiques habituelles de sécurité, pourrait permettre de renforcer de manière non négligeable la sécurité des infrastructures informatiques d'un État. Il faudra en revanche tenir compte du coût financier et de la difficulté de la mise en œuvre.

6 Conclusion

Parmi les trois grands types d'informatique confidentielle, la méthode la plus satisfaisante intellectuellement est celle utilisant le chiffrement homomorphe. En effet celui-ci n'est pas vulnérable, par construction, à la perte de confidentialité des données et il ne requiert la confiance ni dans le fournisseur d'infrastructure ni dans le fabricant du matériel utilisé. En revanche ses performances relativement faibles, et le manque de solution commerciales le rendent, pour le moment, peu attractif. Il en va de même pour le calcul multipartite sécurisé. Cependant les recherches actives dans les deux domaines pourraient conduire à de nouvelles avancées importantes dans les années à venir, atténuant significativement les limites actuelles.

Les TEE semblent donc aujourd'hui les plus matures et prometteurs pour l'informatique confidentielle. La combinaison de composants matériels sécurisés et de techniques d'attestation à distance forme l'assise de ces environnements qui ont pour but de protéger les données sensibles ou secrètes ainsi que le code exécuté contre des attaques de plus en plus fréquentes visant des données en cours de calcul. D'abord ils permettent d'augmenter le niveau de sécurité d'une infrastructure informatique partagée en isolant mieux les applications les unes des autres, empêchant par exemple une application compromise d'accéder aux données d'une autre. Ensuite, ils pourraient permettre d'accroître le niveau de confiance des clients de cette infrastructure pour le traitement de certaines données, pour autant que ces clients puissent garder le contrôle de la gestion des clés de chiffrement et déchiffrement utilisées au sein de l'environnement de confiance.

Pour le client, l'évaluation des risques liés à l'utilisation du système est simplifiée en ce sens que la confiance est, en principe, placée dans un composant matériel spécifique et un micrologiciel vérifiable à distance, tous deux avec un comportement attendu, plutôt que dans l'ensemble d'un ordinateur. Mais d'autres risquent sont introduits, comme, par exemple, la possibilité d'une dépendance accrue à une technique TEE particulière.

Enfin, la protection des données en cours d'utilisation permise par l'informatique confidentielle ne représente qu'un des multiples aspects techniques à considérer concernant la confidentialité des données sensibles (sans compter les aspects juridiques, économiques et politiques). Tout comme la meilleure serrure sur la porte d'entrée principale d'une maison ne résout pas le problème d'une porte secondaire grande ouverte, l'utilisation de l'informatique confidentielle suppose que les données sont effectivement protégées au repos et en mouvement, mais requiert également une formation spécifique des personnes en charge de l'adaptation (plus ou moins importantes en fonction du type d'informatique confidentielle choisie) et du transfert des applications existantes, notamment les analystes, architectes, et programmeurs.

7 Bibliographie

[1] A. Greenberg, « A Peek Into the Toolkit of the Dangerous Triton Hackers », *Wired*, 10 avril 2019. Consulté le: 4 janvier 2023. [En ligne]. Disponible sur: https://www.wired.com/story/triton-hacker-toolkit-fireeye/

[2] K. Zetter, « Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid », *Wired*, 3 mars 2016. Consulté le: 4 janvier 2023. [En ligne]. Disponible sur: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

[3] R. Lakshmanan, « New Malware Families Found Targeting VMware ESXi Hypervisors », *The Hacker News*, 30 septembre 2022. Consulté le: 9 janvier 2023. [En ligne]. Disponible sur: https://thehackernews.com/2022/09/new-malware-families-found-targeting.html

[4] R. Naraine, « Stuxnet attackers used 4 Windows zero-day exploits », *ZDNET*, 14 septembre 2010. Consulté le: 9 janvier 2023. [En ligne]. Disponible sur: https://www.zdnet.com/article/stuxnet-attackers-used-4-windows-zero-day-exploits/

[5] C. Zhao, « SolarWinds, Probably Hacked by Russia, Serves White House, Pentagon, NASA », *Newsweek*, 14 décembre 2020. Consulté le: 9 janvier 2023. [En ligne]. Disponible sur: https://www.newsweek.com/solar-winds-probably-hacked-russia-serves-white-house-pentagon-nasa-1554447

[6] B. Barrett, « Russia's Elite Hackers Have a Clever New Trick That's Very Hard to Fix », *Wired*, 27 septembre 2018. Consulté le: 4 janvier 2023. [En ligne]. Disponible sur: https://www.wired.com/story/fancy-bear-hackers-uefi-rootkit/

[7] T. Bletsch, X. Jiang, V. W. Freeh, et Z. Liang, « Jump-oriented programming: a new class of code-reuse attack », in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, Hong Kong China: ACM, mars 2011, p. 30-40. doi: 10.1145/1966913.1966919.

[8] T. Claburn, « MSI motherboards found to have insecure Secure Boot », *The Register*, 17 janvier 2023. Consulté le: 18 janvier 2023. [En ligne]. Disponible sur: https://www.theregister.com/2023/01/17/msi_motherboards_secure_boot/

[9] C. Gentry, « Fully homomorphic encryption using ideal lattices », in *Proceedings of the 41st annual ACM symposium on Theory of computing*, Bethesda MD USA: ACM, mai 2009, p. 169-178. doi: 10.1145/1536414.1536440.

[10] M. Naehrig, K. Lauter, et V. Vaikuntanathan, « Can homomorphic encryption be practical? », in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11*, Chicago, Illinois, USA: ACM Press, 2011, p. 113. doi: 10.1145/2046660.2046682.

[11] A. C. Yao, « Protocols for Secure Computations », présenté à 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, nov. 1982. doi: 10.1109/SFCS.1982.38.

[12] « TEE System Architecture v1.3 ». GlobalPlatform, mai 2022. [En ligne]. Disponible sur: https://globalplatform.org/specs-library/tee-system-architecture/

[13] M. Sabt, M. Achemlal, et A. Bouabdallah, « Trusted Execution Environment: What It is, and What It is Not », in *2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland: IEEE, août 2015, p. 57-64. doi: 10.1109/Trustcom.2015.357.

[14] N. Drucker et S. Gueron, « Combining Homomorphic Encryption with Trusted Execution Environment: A Demonstration with Paillier Encryption and SGX », in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, Dallas Texas USA: ACM, oct. 2017, p. 85-88. doi: 10.1145/3139923.3139933.

[15] S. Evdokimov et O. Guenther, « Encryption Techniques for Secure Database Outsourcing ». 2007. Consulté le: 20 janvier 2023. [En ligne]. Disponible sur: https://eprint.iacr.org/2007/335

[16] Y. Yang *et al.*, « A Comprehensive Survey on Secure Outsourced Computation and Its Applications », *IEEE Access*, vol. 7, p. 159426-159465, 2019, doi: 10.1109/ACCESS.2019.2949782.

[17] P. Paillier, « Public-Key Cryptosystems Based on Composite Degree Residuosity Classes », in *Advances in Cryptology — EUROCRYPT '99*, J. Stern, Éd., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, p. 223-238. doi: 10.1007/3-540-48910-X_16.

[18] R. Rivest, L. Adleman, et M. Dertouzos, « On Data Banks and Privacy Homomorphisms », *Found. Secure Comput.*, 1978, [En ligne]. Disponible sur: https://pdfs.semanticscholar.org/3c87/22737ef9f37b7a1da6ab81b54224a3c64f72.pdf

[19] « Homomorphic Encryption Standardization – An Open Industry / Government / Academic Consortium to Advance Secure Computation ». https://homomorphicencryption.org/ (consulté le 16 janvier 2023).

[20] A. Acar, H. Aksu, A. S. Uluagac, et M. Conti, « A Survey on Homomorphic Encryption Schemes: Theory and Implementation », *ACM Comput. Surv.*, vol. 51, nº 4, p. 1-35, juill. 2018, doi: 10.1145/3214303.

[21] C. Moore, M. O'Neill, E. O'Sullivan, Y. Doroz, et B. Sunar, « Practical homomorphic encryption: A survey », in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, Melbourne VIC, Australia: IEEE, juin 2014, p. 2792-2795. doi: 10.1109/ISCAS.2014.6865753.

[22] S. Goldwasser et S. Micali, « Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information », *J. Comput. Syst. Sci.*, vol. 28, nº 2, p. 270-299, avr. 1984, doi: 10.1016/0022-0000(84)90070-9.

[23] P. Y. A. Ryan, « Prêt à Voter with Paillier encryption », *Math. Comput. Model.*, vol. 48, nº 9-10, p. 1646-1662, nov. 2008, doi: 10.1016/j.mcm.2008.05.015.

[24] M. Joye et F. A. P. Petitcolas, « PINFER: Privacy-Preserving Inference Logistic Regression, Support Vector Machines, and More, Over Encrypted Data », 2019.

[25] R. A. Popa, C. M. S. Redfield, N. Zeldovich, et H. Balakrishnan, « CryptDB: protecting confidentiality with encrypted query processing », in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, Cascais Portugal: ACM, oct. 2011, p. 85-100. doi: 10.1145/2043556.2043566.

[26] C. Jost, H. Lam, A. Maximov, et B. Smeets, « Encryption Performance Improvements of the Paillier Cryptosystem ». Cryptology ePrint Archive, 2015. [En ligne]. Disponible sur: https://eprint.iacr.org/2015/864

[27] C. Gentry et S. Halevi, « Implementing Gentry's Fully-Homomorphic Encryption Scheme », in *Advances in Cryptology – EUROCRYPT 2011*, K. G. Paterson, Éd., in Lecture Notes in Computer Science, vol. 6632. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, p. 129-148. doi: 10.1007/978-3-642-20465-4_9.

[28] Z. Brakerski, C. Gentry, et V. Vaikuntanathan, « Fully Homomorphic Encryption without Bootstrapping ».

[29] « HElib ». homenc, 1 mars 2023. Consulté le: 1 mars 2023. [En ligne]. Disponible sur: https://github.com/homenc/HElib

[30] I. Chillotti, N. Gama, M. Georgieva, et M. Izabachène, « TFHE: Fast Fully Homomorphic Encryption Over the Torus », *J. Cryptol.*, vol. 33, nº 1, p. 34-91, janv. 2020, doi: 10.1007/s00145-019-09319-x.

[31] « Microsoft SEAL ». Microsoft, 1 mars 2023. Consulté le: 1 mars 2023. [En ligne]. Disponible sur: https://github.com/microsoft/SEAL

[32] J. Fan et F. Vercauteren, « Somewhat Practical Fully Homomorphic Encryption ».

[33] C. Gentry, « Fully Homomorphic Encryption: current State of the Art », 2017. [En ligne]. Disponible sur: https://www.maths.ox.ac.uk/system/files/attachments/FHE1.pptx

[34] « TFHE Fast Fully Homomorphic Encryption over the Torus ». https://tfhe.github.io/tfhe/ (consulté le 1 mars 2023).

[35] A. Al Badawi *et al.*, « OpenFHE: Open-Source Fully Homomorphic Encryption Library », in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, Los Angeles CA USA: ACM, nov. 2022, p. 53-63. doi: 10.1145/3560827.3563379.

[36] « Garbled circuit », *Wikipedia*. 9 décembre 2022. Consulté le: 16 janvier 2023. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Garbled_circuit&oldid=1126505060

[37] N. P. Smart, « Secure MPC Based on Secret Sharing », 31 août 2018. Consulté le: 15 juin 2023. [En ligne]. Disponible sur: https://homes.esat.kuleuven.be/~nsmart/FHE-MPC/LSSS-MPC.pdf

[38] K. Verslype, « Secure multiparty computation – Collectieve berekeningen op verspreide gevoelige gegevens | Smals Research », 8 juin 2021. https://www.smalsresearch.be/secure-multiparty-computation-collectieve-berekeningen-op-verspreide-gevoelige-gegevens/ (consulté le 6 février 2023).

[39] S. Checkoway et H. Shacham, « Iago Attacks: Why the System Call API is a Bad Untrusted RPC Interface ».

[40] « Common Criteria for Information Technology Security Evaluation ». novembre 2022. [En ligne]. Disponible sur: https://www.commoncriteriaportal.org/cc/

[41] « TEE Protection Profile - Version 1.3 ». GlobalPlatform, 4 juin 2020.

[42] « A Technical Analysis of Confidential Computing ». Confidential Computing Consortium, octobre 2021. [En ligne]. Disponible sur: https://confidential-

computing.io/wp-content/uploads/sites/85/2022/11/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.2_updated_2022-11-02.pdf

[43] M. Pei, H. Tschofenig, D. Thaler, et D. Wheeler, « Trusted Execution Environment Provisioning (TEEP) Architecture », Internet Engineering Task Force, Internet Draft draft-ietf-teep-architecture-19, oct. 2022. Consulté le: 30 janvier 2023. [En ligne]. Disponible sur: https://datatracker.ietf.org/doc/draft-ietf-teep-architecture-19

[44] D. A. Cooper, W. T. Polk, A. R. Regenscheid, et M. P. Souppaya, « BIOS protection guidelines », National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-147, 2011. doi: 10.6028/NIST.SP.800-147.

[45] D. Challener et K. Goldman, « Trusted Platform Module (TPM) », *Trusted Computing Group (TCG)*, 2019. https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[46] E. Barker et W. C. Barker, « Recommendation for key management:: part 2 -- best practices for key management organizations », National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-57pt2r1, mai 2019. doi: 10.6028/NIST.SP.800-57pt2r1.

[47] « Trusted Platform Module Library Part 1: Architecture ». 8 novembre 2019. Consulté le: 16 janvier 2023. [En ligne]. Disponible sur: https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf

[48] National Institute of Standards and Technology, « Security requirements for cryptographic modules », National Institute of Standards and Technology, Gaithersburg, MD, NIST FIPS 140-3, avr. 2019. doi: 10.6028/NIST.FIPS.140-3.

[49] G. Coker *et al.*, « Principles of remote attestation », *Int. J. Inf. Secur.*, vol. 10, nº 2, p. 63-81, juin 2011, doi: 10.1007/s10207-011-0124-7.

[50] « Nix & NixOS | Reproducible builds and deployments ». https://nixos.org/ (consulté le 6 juin 2023).

[51] H. Birkholz, D. Thaler, M. Richardson, N. Smith, et W. Pan, « Remote ATtestation procedureS (RATS) Architecture », Internet Engineering Task Force, Request for Comments RFC 9334, janv. 2023. doi: 10.17487/RFC9334.

[52] M. U. Sardar et C. Fetzer, « Confidential computing and related technologies: a critical review », *Cybersecurity*, vol. 6, nº 1, p. 10, mai 2023, doi: 10.1186/s42400-023-00144-1.

[53] « Cryptographic Algorithm Recommendations ». GlobalPlatform, juin 2021. [En ligne]. Disponible sur: https://globalplatform.org/specs-library/globalplatform-technology-cryptographic-algorithm-recommendations/

[54] K. Alshmrany *et al.*, « Position Paper: Towards a Hybrid Approach to Protect Against Memory Safety Vulnerabilities », in *2022 IEEE Secure Development Conference (SecDev)*, Atlanta, GA, USA: IEEE, oct. 2022, p. 52-58. doi: 10.1109/SecDev53368.2022.00020.

[55] L. Szekeres, M. Payer, Tao Wei, et D. Song, « SoK: Eternal War in Memory », in *2013 IEEE Symposium on Security and Privacy*, Berkeley, CA: IEEE, mai 2013, p. 48-62. doi: 10.1109/SP.2013.13.

[56] « Microsoft: 70 percent of all security bugs are memory safety issues », *ZDNET*. https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues/ (consulté le 18 janvier 2023).

[57] M. Abadi, M. Budiu, et J. Ligatti, « Control-Flow Integiry - Principles, Implementations, and Applications ».

[58] L. H. Newman, « As Chips Shrink, Rowhammer Attacks Get Harder to Stop », *Wired*, 26 mai 2021. Consulté le: 14 juin 2023. [En ligne]. Disponible sur: https://www.wired.com/story/rowhammer-half-double-attack-bit-flips/

[59] J. True et N. Asadizanjani, « Physical Inspection and Attacks on Electronics: An Academic Course for the Hardware Cybersecurity Workforce », *IEEE Secur. Priv.*, vol. 21, nº 2, p. 63-69, mars 2023, doi: 10.1109/MSEC.2023.3236999.

[60] Y. Zhou et D. Feng, « Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing ». Cryptology ePrint Archive, 2005. [En ligne]. Disponible sur: https://eprint.iacr.org/2005/388

[61] « Software Guard Extensions - Wikipedia ». https://en.wikipedia.org/wiki/Software_Guard_Extensions#List_of_SGX_vulnerabilities (consulté le 18 janvier 2023).

[62] « Security Best Practices for Side Channel Resistance », *Intel*, 15 mars 2019. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/secure-coding/security-

best-practices-side-channel-resistance.html (consulté le 18 janvier 2023).

[63] D. Lee, D. Jung, I. T. Fang, C.-C. Tsai, et R. A. Popa, « An Off-Chip Attack on Hardware Enclaves via the Memory Bus ».

[64] O. Goldreich, « Towards a theory of software protection and simulation by oblivious RAMs », in *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*, New York, New York, United States: ACM Press, 1987, p. 182-194. doi: 10.1145/28395.28416.

[65] G. Asharov, I. Komargodski, W.-K. Lin, K. Nayak, et E. Peserico, « OptORAMa: Optimal Oblivious RAM », présenté à EUROCRYPT 2020, IACR, nov. 2020.

[66] D. Kaplan, « AMD x86 Memory Encryption Technologies », août 2016. Consulté le: 11 mai 2023. [En ligne]. Disponible sur: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kaplan

[67] D. Kaplan, J. Powell, et T. Woller, « AMD Memory Encryption », White Paper, oct. 2021. Consulté le: 1 mai 2023. [En ligne]. Disponible sur: https://www.amd.com/system/files/TechDocs/memory-encryption-white-paper.pdf

[68] « AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More », AMD, janv. 2020.

[69] « Secure Encrypted Virtualisation API Version 0.24 », Specification, avr. 2020. [En ligne]. Disponible sur: https://www.amd.com/system/files/TechDocs/55766_SEV-KM_API_Specification.pdf

[70] Victor Costan et Srinivas Devadas, « Intel SGX Explained ». Cryptology ePrint Archive, 2016. [En ligne]. Disponible sur: https://eprint.iacr.org/2016/086

[71] P.-C. Cheng *et al.*, « Intel TDX Demystified: A Top-Down Approach ». arXiv, 27 mars 2023. Consulté le: 23 mai 2023. [En ligne]. Disponible sur: http://arxiv.org/abs/2303.15540

[72] « Intel 64 and IA-32 Architectures Software Developer's Manual, Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4 ». Intel Corporation, mars 2023. [En ligne]. Disponible sur: https://cdrdv2.intel.com/v1/dl/getContent/671200

[73] A. Rao, « Rising to the Challenge — Data Security with Intel Confidential Computing », 20 janvier 2022. https://community.intel.com/t5/Blogs/Products-and-Solutions/Security/Rising-to-the-Challenge-Data-Security-with-Intel-Confidential/post/1353141 (consulté le 17 mai 2023).

[74] G. Steer, « Finance's big tech problem », *Financial Times*, 6 juillet 2022. Consulté le: 10 juillet 2023. [En ligne]. Disponible sur: https://www.ft.com/content/41f400b6-f83f-4fa1-8dac-731acddcf8f2

[75] « The Security Design of the AWS Nitro System - AWS Whitepaper ». AWS, 18 novembre 2022. [En ligne]. Disponible sur: https://docs.aws.amazon.com/fr_fr/whitepapers/latest/security-design-of-aws-nitro-system/security-design-of-aws-nitro-system.html

[76] « Confidential computing: an AWS perspective | AWS Security Blog », 24 août 2021. https://aws.amazon.com/blogs/security/confidential-computing-an-aws-perspective/ (consulté le 18 avril 2023).

[77] https://techcommunity.microsoft.com/t5/user/viewprofilepage/user-id/1182566, « Preview: Introducing DCesv5 and ECesv5-series Confidential VMs with Intel TDX », *TECHCOMMUNITY.MICROSOFT.COM*, 24 avril 2023. https://techcommunity.microsoft.com/t5/azure-confidential-computing/preview-introducing-dcesv5-and-ecesv5-series-confidential-vms/ba-p/3800718 (consulté le 16 mai 2023).

[78] « Advanced Trusted Environment: OMTP TR1 », Open Mobile Terminal Platform (OMTP), mai 2009. [En ligne]. Disponible sur: file:///C:/Users/fape/Downloads/OMTP_Advanced_Trusted_Environment_OMTP_TR1_v1_1.pdf

[79] R. Koppel et C. Kuziemsky, « Healthcare Data Are Remarkably Vulnerable to Hacking: Connected Healthcare Delivery Increases the Risks », *Stud. Health Technol. Inf.*, vol. 257, p. 218-222, 2019.

[80] C. Shepherd *et al.*, « Secure and Trusted Execution: Past, Present, and Future - A Critical Review in the Context of the Internet of Things and Cyber-Physical Systems », in *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China: IEEE, août 2016, p. 168-177. doi: 10.1109/TrustCom.2016.0060.

[81] R. A. Popa, « Secure computation: Homomorphic encryption or hardware enclaves? », *MC2 Project*, 28 septembre 2021. https://medium.com/mc2-project/secure-computation-homomorphic-encryption-or-hardware-enclaves-83da90102593 (consulté le 20 janvier 2023).

[82] « UN Handbook for Privacy-Preserving Computation Techniques », 2023. [En ligne]. Disponible sur: https://unstats.un.org/bigdata/task-teams/privacy/UN%20Handbook%20for%20Privacy-Preserving%20Techniques.pdf

[83] Jeremy Kun, « Google's Fully Homomorphic Encryption Compiler — A Primer », *Math ∩ Programming*, 13 février 2023. https://jeremykun.com/2023/02/13/googles-fully-homomorphic-encryption-compiler-a-primer/ (consulté le 28 février 2023).

[84] D. Escudero, « An Introduction to Secret-Sharing-Based Secure Multiparty Computation ». 4 juin 2023. [En ligne]. Disponible sur: https://ia.cr/2022/062

[85] X. Zhao, M. Li, E. Feng, et Y. Xia, « Towards A Secure Joint Cloud With Confidential Computing », in *2022 IEEE International Conference on Joint Cloud Computing (JCC)*, Fremont, CA, USA: IEEE, août 2022, p. 79-88. doi: 10.1109/JCC56315.2022.00019.

[86] R. Frank, « G-Cloud Voortgangsrapport », oct. 2020. [En ligne]. Disponible sur: https://www.frankrobben.be/wp-content/uploads/2020/11/G-Cloud-Voortgangsrapport-algemeen-oktober-2020.pdf

Fabien A. P. Petitcolas est consultant en recherche chez Smals où il se concentre sur les questions de sécurité. Avant de rejoindre Smals, Fabien a occupé différents postes chez OneSpan et Microsoft. Diplômé de l'École centrale de Lyon, Fabien a ensuite obtenu son doctorat (PhD) à l'université de Cambridge sous la direction du professeur Ross Anderson FRS FREng. Contact : fabien.petitcolas@smals.be.

Au sujet de Smals et Smals Research

Smals fournit des services ICT en fonction des besoins spécifiques du secteur public et des dernières évolutions technologiques. Avec sa propre équipe de chercheurs, munis d'un bagage académique de taille, généralement de niveau 'doctorat', Smals investit dans la recherche et le développement consacrés à divers domaines technologiques, soigneusement sélectionnés en concertation avec les clients-membres.

Le département Recherche de Smals suit constamment les dernières évolutions du marché et teste les nouveaux concepts et techniques sur leur faisabilité pratique pour ensuite faire part de ses constats aux clients-membres et aux équipes internes de Smals. Une attention particulière est accordée à l'adoption des techniques et concepts que l'équipe propose.