

LE LIVRE BLANC DE “VALIDY TECHNOLOGY”

SOLUTION CONTRE LE PIRATAGE DES LOGICIELS
ET LE SABOTAGE INFORMATIQUE



SÉCURITÉ INFORMATIQUE ET ÉCONOMIQUE

www.validy.com / www.validy-licensing.com

SOMMAIRE

page 1	1. Les mesures actuelles contre le piratage des logiciels
	1.1 Protections additives simples
	1.2 Protections additives avec dispositif matériel
	1.3 Protections soustractives
page 2	2. Les mesures actuelles contre le sabotage informatique
	2.1 Vérification de l'intégrité avant le démarrage de l'exécution
	2.2 Vérification de l'intégrité avec une exécution à l'intérieur d'un dispositif matériel
	2.3 Vérification de l'intégrité par exécution parallèle
page 3	3. L'invention Validy
	3.1 Les quatre critères pour une méthode soustractive qui fonctionne
	3.2 Le système de protection de l'exécution qui répond à ces quatre critères
page 6	4. Importants bénéfices additionnels
	4.1 Système d'identification d'instructions "tag"
	4.2 Protection mutuelle
	4.3 Calculs additifs, calculs soustractifs
	4.4 Verrouillage des calculs additifs
	4.5 Rétorsion ultime
	4.6 Intégrité du programme
	4.7 Le système de protection de données
	4.8 Mesures d'usage
	4.9 Limitations de fonctionnalités
	4.10 Rigidité ou flexibilité du dispositif matériel
	4.11 Dispositif matériel fixe ou amovible
page 10	5. Développement de programme
	5.1 Implications de l'utilisation de "Validy Technology" dans le développement
	5.2 Protection contre le piratage, pas de librairie de sécurité
	5.3 Processus de debug
	5.4 Protection de l'intégrité
	5.5 Compilateur Validy
	5.6 Autres compilateurs
	5.7 Outils additionnels
page 11	6. Implications au niveau déploiement
	6.1 Les différents supports
	6.2 La production
	6.3 La maintenance

I • LES MESURES ACTUELLES CONTRE LE PIRATAGE DES LOGICIELS

P

I•1 Protections additives simples

Les protections additives consistent à vérifier les droits d'exécution par exemple en testant la présence d'un CD-ROM, ou en testant sur l'ordinateur un n° de série, ou en testant sur Internet un n° de série, ...

Ces tests sont rajoutés au programme par le développeur. Ils ne sont pas indispensables au fonctionnement du programme. Ils peuvent être facilement repérés et compris par le pirate. Celui-ci peut alors soit répondre correctement soit les court-circuiter.

Afin de rendre la tâche plus difficile au pirate, le développeur utilise parfois des techniques dites "d'obscurcissement" qui essayent de cacher ces ajouts.

Malgré ces techniques, le pirate déterminé finit toujours par trouver et contourner la protection. Il suffit qu'un seul pirate publie sur Internet ces résultats sous forme d'un programme automatique (appelé crack) pour que n'importe quel utilisateur d'Internet soit à son tour capable de contourner la protection additive.

I•2 Protections additives avec dispositif matériel

Afin de renforcer la protection, certaines sociétés ont conçu des dispositifs matériels appelés "dongles".

Ces dispositifs peuvent prendre des formes différentes : bouchon sur le port parallèle de l'ordinateur, carte à puce, clef USB, circuit électronique, ... Ceux-ci utilisent souvent des techniques de cryptographie qui empêchent le pirate de répondre correctement aux tests.

Malgré cela, le pirate, même s'il ne sait pas répondre correctement aux tests, finit toujours par arriver à les court-circuiter.

I•3 Protections soustractives

Les protections soustractives consistent à cacher une partie du programme dans un dispositif matériel afin de soustraire cette partie de la vue du pirate. Ce dispositif matériel peut aussi prendre différentes formes : bouchon sur le port parallèle de l'ordinateur, carte à puce, clef USB, circuit électronique, ...

Les protections soustractives classiques consistent à déporter des fonctions dans le dispositif matériel. Le processus se déroule de la manière suivante :

Les paramètres d'une fonction sont fournis au dispositif matériel par le programme protégé, la fonction est exécutée par le dispositif matériel et enfin les résultats de cette fonction sont retournés au programme protégé (système de "Remote Procedure Call").

La difficulté est alors de trouver les "bonnes" fonctions à déporter dans le dispositif matériel. Le pirate dispose en effet de plusieurs méthodes pour contourner la protection :

Si la fonction déportée n'est pas indispensable à l'exécution du programme, le pirate peut simplement la supprimer tout en gardant l'essentiel de la fonctionnalité du programme.

Si la fonction déportée utilise souvent les mêmes valeurs pour ses paramètres, le pirate après avoir analysé pendant un certain temps les échanges entre le programme protégé et le dispositif matériel est capable d'écrire un programme qui imite le dispositif matériel et répond à sa place, même s'il ne comprend pas la fonction déportée.

Si la fonction déportée est identifiable, le pirate après avoir analysé pendant un certain temps les échanges entre le programme protégé et le dispositif matériel est capable d'écrire un programme qui imite le dispositif matériel et simule la fonction déportée quels que soient les paramètres.

Dans les trois cas ci-dessus, avec un peu de ténacité, le pirate est alors capable de modifier suffisamment le programme protégé pour qu'il fonctionne sans le dispositif matériel.

Pour palier à ces faiblesses, le développeur est alors amené à déporter si possible des fonctions compliquées. Cette pratique est limitée pour deux raisons :

Beaucoup de programmes ne comportent pas de fonctions suffisamment compliquées.

Si elles existent, l'exécution de ces fonctions compliquées dans le dispositif matériel, qui est beaucoup plus lent que l'ordinateur, ralentit considérablement le programme protégé ce qui devient vite rédhibitoire.

2•LES MESURES ACTUELLES CONTRE LE SABOTAGE INFORMATIQUE



2•1 Vérification de l'intégrité avant le démarrage de l'exécution

La vérification de l'intégrité d'un programme avant son démarrage consiste à utiliser un programme auxiliaire qui vérifie que le programme à exécuter n'a pas été modifié.

Cette vérification possède deux faiblesses exploitables par un pirate :

Le pirate peut modifier le programme après la vérification et avant l'exécution.

Le pirate peut modifier le programme de vérification auxiliaire pour qu'il accepte des programmes modifiés.

2•2 Vérification de l'intégrité avec une exécution à l'intérieur d'un dispositif matériel

Quand un programme est exécuté entièrement à l'intérieur d'un dispositif matériel, le dispositif peut vérifier lui-même de manière sûre l'intégrité du programme avant son démarrage et les deux faiblesses du point précédent sont éliminées. Cette protection est très efficace mais limite considérablement la taille et la performance du programme protégé qui doit maintenant s'exécuter entièrement dans le dispositif matériel, beaucoup plus petit et plus lent que l'ordinateur (ralentissement d'un facteur 100 au moins).

2•3 Vérification de l'intégrité par exécution parallèle

Quand une très grande fiabilité est requise, un même programme ou des programmes similaires peuvent être exécutés en parallèle sur plusieurs machines. De manière régulière, les différentes machines vérifient mutuellement que le déroulement des programmes est similaire et, en cas de problème, une décision de correction est prise. Toute modification de l'intégrité du programme nécessite donc un accès à l'ensemble des machines et une modification de l'ensemble des programmes. Ces approches très complexes et très onéreuses sont réservées à des applications très particulières (spatiales, nucléaires, ...).

3 • L'INVENTION VALIDY



3•1 Les quatre critères pour une méthode soustractive qui fonctionne

Comme déjà abordé dans le paragraphe décrivant les protections soustractives, la partie du programme cachée dans un dispositif matériel doit répondre à un certain nombre de critères. Pour que la protection soit effectivement efficace et utilisable, quatre critères fondamentaux doivent être respectés :

- **La partie cachée doit être essentielle :**

Il ne faut pas rajouter une partie de programme qui soit uniquement "décorative" dans le but de la cacher dans le dispositif matériel. En effet, le pirate peut dans ce cas simplement la supprimer tout en gardant l'essentiel de la fonctionnalité du programme.

Par exemple, si la partie cachée du programme sert uniquement à calculer un arrière-plan de fenêtre, le programme gardera l'essentiel de sa fonctionnalité même si l'arrière-plan ne s'affiche pas.

- **La partie cachée doit être non simulable :**

En analysant les échanges entre le programme protégé et le dispositif matériel, le pirate ne doit pas être capable de comprendre le fonctionnement de la partie du programme qui s'exécute à l'intérieur du dispositif matériel. En effet, une telle compréhension lui permettrait d'écrire un programme de simulation pour s'affranchir du dispositif matériel.

Par exemple, si la partie cachée exécute une fonction sinus, le pirate peut l'identifier et la simuler.

- **La partie cachée doit être non rejouable :**

En analysant pendant un certain temps les échanges entre le programme protégé et le dispositif matériel, le pirate ne doit pas être capable d'imiter les réponses du dispositif matériel. En effet, une telle imitation, même sans comprendre le sens des dialogues, lui permettrait d'écrire un programme d'imitation pour s'affranchir du dispositif matériel.

Par exemple, si le dialogue entre le programme protégé et le dispositif matériel se répète d'une exécution à l'autre, le pirate peut enregistrer les réponses lors d'une première exécution puis les "rejouer" avec son programme d'imitation lors des exécutions suivantes.

- **La partie cachée ne doit pas ralentir l'exécution :**

Pour qu'une protection soustractive soit utilisable, il faut que le ralentissement qu'elle introduit dans l'exécution du programme protégé soit suffisamment faible.

En effet, la différence de performance entre le processeur exécutant la partie non cachée du programme protégé et celle du dispositif matériel exécutant la partie cachée du programme protégé est souvent très importante (de l'ordre de 100 ou plus). Le choix judicieux de la partie cachée est donc primordial.

3 • L'INVENTION VALIDY



3•2 Le système de protection de l'exécution qui répond à ces quatre critères

Les six facettes de l'invention Validy :

• Variable :

La première facette de l'invention consiste à faire résider des variables dans le dispositif matériel pendant l'exécution d'un programme protégé.

Lors de l'affectation d'une valeur à une variable résidant dans le dispositif matériel, cette valeur est transférée vers le dispositif matériel.

Lors de l'utilisation d'une variable résidant dans le dispositif matériel, la valeur de celle-ci est transférée vers l'ordinateur.

Pendant leur séjour dans le dispositif matériel, les valeurs des variables ne sont pas visibles par un pirate. Cette facette introduit une protection par le fait qu'une partie de l'état du programme réside dans le dispositif matériel et que par conséquent, la présence de celui-ci est indispensable au fonctionnement du programme.

Cependant cette facette utilisée seule n'est pas suffisante pour assurer une protection efficace car un pirate déterminé peut observer et comprendre le dialogue entre l'ordinateur et le dispositif matériel et réaliser un simulateur de ce dernier.

• Dissociation temporelle :

La seconde facette de l'invention consiste à modifier la valeur des variables déportées dans le dispositif matériel. Ces modifications sont effectuées indépendamment de l'instant des transferts depuis ou vers le dispositif matériel, elles sont déclenchées par des commandes envoyées de l'ordinateur vers le dispositif matériel.

Cette facette renforce la protection en complexifiant le dialogue entre l'ordinateur et le dispositif matériel par la décorrélation des transferts et des commandes de modification de chaque variable et des variables entre elles.

Cependant ceci n'est toujours pas suffisant pour assurer une protection efficace, car malgré la complexification du dialogue, un pirate déterminé peut toujours observer, modifier et comprendre le dialogue entre l'ordinateur et le dispositif matériel et réaliser un simulateur de ce dernier.

• Fonctions élémentaires :

La troisième facette de l'invention consiste à décomposer les fonctions modifiant les variables à l'intérieur du dispositif matériel en fonctions élémentaires. Ces fonctions élémentaires sont déclenchées par des commandes élémentaires envoyées de l'ordinateur vers le dispositif matériel. Ces fonctions élémentaires sont du type ADD, MOVE, ...

Cette facette permet de standardiser le dialogue par un jeu de commandes élémentaires qui déclenchent des fonctions élémentaires à l'intérieur du dispositif matériel afin de synthétiser n'importe quelle fonction.

Cette facette renforce encore la protection car il n'existe plus de limitation ni du nombre ni de la taille des fonctions pouvant être traitées à l'intérieur du dispositif matériel. De plus, les commandes élémentaires permettant de synthétiser des fonctions différentes peuvent être entremêlées pour complexifier le dialogue.

Cependant ceci n'est toujours pas suffisant pour assurer une protection efficace car malgré l'augmentation de la complexité du dialogue, un pirate déterminé peut toujours observer, modifier et finir par comprendre la signification des commandes élémentaires et réaliser un simulateur.

3 • L'INVENTION VALIDY

R

- **Renommage :**

La quatrième facette de l'invention consiste à renommer les commandes élémentaires échangées entre l'ordinateur et le dispositif matériel de manière à les rendre inintelligibles. Ce renommage est effectué en encryptant les commandes élémentaires lors de la protection du programme. Lors de l'exécution, le dispositif matériel décrypte les commandes élémentaires renommées qu'il reçoit et exécute les fonctions élémentaires déclenchées.

Cette facette renforce la protection, car de cette façon le pirate n'a jamais accès aux commandes élémentaires non cryptées et ne peut donc pas analyser leur structure. Il ne peut pas non plus générer de nouvelles commandes élémentaires pour solliciter le dispositif matériel et essayer de comprendre la réponse à cette sollicitation.

A ce niveau, la protection est déjà très solide et au-delà de la capacité de la plupart des pirates. La dernière attaque possible consiste à essayer d'appréhender les fonctions élémentaires exécutées dans le dispositif matériel en faisant des hypothèses sur leur signification et en testant ces hypothèses en modifiant l'ordre d'exécution de ces fonctions.

- **Détection et coercition :**

La cinquième facette de l'invention consiste à vérifier que la partie du programme exécutée dans le dispositif matériel est exécutée de manière conforme à ce qui avait été prévu lors de la phase de protection du programme (compilation). A cette fin, le dispositif matériel doit être considéré comme un coprocesseur de sécurité exécutant un jeu d'instructions de sécurité. Ces instructions de sécurité comportent des champs supplémentaires permettant de contrôler que l'enchaînement de leurs exécutions est conforme à ce qui avait été prévu. Si l'enchaînement n'est pas conforme, le coprocesseur de sécurité le détecte immédiatement et prend les mesures de coercition décidées par le programmeur. Comme le programme protégé ne peut fonctionner sans le coprocesseur, celui-ci possède un réel moyen de rétorsion puisqu'il peut décider de s'arrêter de fonctionner et ainsi stopper le programme.

Cette facette interdit la dernière attaque possible consistant à modifier l'ordre d'exécution des instructions pour essayer de les comprendre.

A ce niveau, le pirate ne peut plus toucher aux échanges entre l'ordinateur et le dispositif matériel sans être immédiatement détecté.

- **Branchement conditionnel :**

La sixième et dernière facette de l'invention consiste à masquer la structure du programme en se servant du dispositif matériel pour fournir des adresses de branchement.

Pour cela il faut faire résider dans le dispositif matériel des variables d'état qui conditionnent l'exécution du programme, effectuer dans le dispositif matériel les calculs permettant de connaître l'adresse de branchement et enfin retourner cette adresse à l'ordinateur.

Cette facette parachève la protection en cachant de la vue du pirate les décisions et les raisons des décisions de branchement ce qui lui enlève la logique d'implémentation du programme.

4•IMPORTANTS BÉNÉFICES ADDITIONNELS

S

4•1 Système d'identification d'instructions "tag"

La manière la plus efficace d'implémenter l'invention détection et coercition est d'utiliser un système d'identification d'instructions utilisant des étiquettes appelées "tags".

Ces "tags" font partie intégrante du code opération de chaque instruction et des ressources du coprocesseur.

Dans chaque instruction, il y a un tag pour identifier cette instruction et un ou plusieurs tags pour identifier la source attendue de chaque opérande. Les registres du coprocesseur permettent non seulement de stocker des données, mais aussi le tag identifiant l'instruction qui a généré la donnée.

Avec ce système d'identification, le coprocesseur est capable de vérifier lors de l'exécution du programme protégé que l'enchaînement de ses instructions n'a pas été modifié.

Une instruction peut être par exemple de la forme :

```
ADD R1 <t1> R2 <t2> R3 <t3>
```

Ce qui signifie : vérifier que R2 a été produit par une instruction identifiée par le tag t2 et que R3 a été produit par une instruction identifiée par le tag t3. Indiquer une erreur éventuelle au système de coercition. Additionner les données de R2 et R3 et stocker le résultat dans la partie donnée de R1. Stocker aussi la valeur t1 comme identifiant d'instruction dans la partie tag du registre R1.

Tout compilateur possède des informations de dépendance permettant de calculer facilement les tags nécessaires pour cette invention.

Dans la phase de protection du programme, le compilateur Validy génère ainsi automatiquement les tags et les insère dans les instructions du coprocesseur de sécurité.

4•2 Protection mutuelle

Le système de détection et coercition permet très simplement de faire dépendre le succès de l'exécution d'un calcul effectué dans le dispositif matériel de l'exécution préalable d'un autre calcul dans le dispositif matériel.

La méthode utilisée pour créer cette dépendance est d'exécuter les deux instructions de sécurité suivantes :

- générer la valeur 0 en soustrayant le résultat du premier calcul à lui-même.
- additionner ce 0 au résultat du second calcul.

Le système de vérification d'enchaînement explicité ci-dessus fait alors échouer le second calcul si le premier calcul n'a pas été réalisé ou si les deux instructions de sécurité ajoutées ont été modifiées ou supprimées.

4•IMPORTANTS BÉNÉFICES ADDITIONNELS

P

Plus précisément :

Supposons que R1 <t1> soit le résultat de calcul1 avec le tag t1 et R2 <t2> soit le résultat de calcul2 avec le tag t2.

et effectuons :

SUB R3 <tempt> R1 <t1> R1 <t1>

ADD R2 <newt2> R2 <t2> R3 < tempt >

La première instruction génère un 0 tagué avec tempt dans le registre R3 si calcul1 a été effectué, et échoue sinon.

La deuxième instruction génère un nouveau tag pour R2 sans changer sa donnée, et échoue si la première instruction a été modifiée ou supprimée.

Au final, le résultat de calcul2 est disponible dans R2 avec le nouveau tag newt2, uniquement si calcul1 a bien été effectué.

Conséquences :

Ce système de dépendances, appliqué plusieurs fois, permet la protection mutuelle de 2 calculs ou plus.

4•3 Calculs additifs, calculs soustractifs

On appelle calcul additif, un calcul qui est rajouté au programme à protéger et qui est effectué dans le dispositif matériel. Puisque c'est un ajout au programme, un calcul additif est par conséquent non indispensable.

On appelle calcul soustractif, un calcul qui appartient au programme à protéger et qui est soustrait de la vue du pirate en l'effectuant dans le dispositif matériel. Puisqu'il fait partie intégrante du programme, un calcul soustractif est par conséquent indispensable.

4•4 Verrouillage des calculs additifs

La protection mutuelle décrite ci-dessus permet de lier entre eux des calculs qu'ils soient soustractifs ou additifs.

Il est alors possible de rajouter au programme des calculs additifs et de créer une toile de liens qui s'appuie sur un "socle" de calculs soustractifs et qui relie tous les calculs additifs à ce socle.

4•5 Rétorsion ultime

Quand tous les calculs sont reliés, toute tentative de modification ou de suppression de l'un quelconque ou de plusieurs des calculs est immédiatement détectée par le dispositif matériel qui prend alors les mesures de coercition appropriées.

La seule solution pour le pirate consiste alors à ôter simultanément l'ensemble de tous les calculs effectués à l'intérieur du dispositif matériel. L'existence d'un "socle" de calculs soustractifs inconnus du pirate et indispensables au programme l'oblige à réinventer l'ensemble de ces calculs simultanément, ce qui est impossible!

4•IMPORTANTS BÉNÉFICES ADDITIONNELS

4•6 Intégrité du programme

Les calculs additifs ajoutés peuvent servir à vérifier le bon fonctionnement du programme et ainsi défendre son intégrité.

Ces calculs ne demandent jamais de transfert d'information du dispositif matériel vers l'ordinateur et sont donc très performants, ils peuvent prendre la forme par exemple de :

- vérification du graphe d'appel d'un sous-programme
- vérification du nombre d'itérations d'une boucle
- vérification du temps écoulé pour exécuter une fonction, pour empêcher l'exécution pas à pas
- etc.

4•7 Le système de protection de données

Les calculs additifs ajoutés peuvent servir à protéger les données du programme et en particulier ses données d'échange.

Par exemple, écriture de données encryptées dans une base de données avec impossibilité pour le pirate de compromettre l'écriture ou la lecture de ces données, et impossibilité de modifier les données de la base.

4•8 Mesures d'usage

Les calculs additifs ajoutés peuvent servir à mesurer l'usage de différentes fonctionnalités du programme telles que par exemple le temps d'utilisation, le nombre de pages imprimées, le nombre de sauvegardes, ... et permet ainsi d'interdire l'usage de celles-ci à partir d'un certain seuil. Ceci permet la vente du logiciel à l'usage (pay per use).

4•9 Limitations de fonctionnalités

Les calculs additifs ajoutés peuvent servir à limiter les fonctionnalités du programme en fonction de l'état du dispositif matériel.

Par exemple :

Selon l'état du dispositif matériel, le même programme servira de version d'évaluation à fonctionnalités réduites ou de version commerciale à pleines fonctionnalités.

4•IMPORTANTS BÉNÉFICES ADDITIONNELS

R

4•10 Rigidité ou flexibilité du dispositif matériel

Selon ses applications, un éditeur de logiciels peut décider d'utiliser un dispositif matériel rigide ou flexible.

Si le dispositif matériel utilisé est rigide, aucune mise à jour du logiciel n'est possible sans changement du dispositif matériel. Ceci permet de s'assurer qu'une version certifiée d'un logiciel n'est plus jamais modifiée. Dans ce cas le dispositif matériel peut être considéré comme un scellé sur le logiciel. Par exemple un logiciel servant au contrôle des freins d'une voiture.

Si le dispositif matériel utilisé est flexible, les mises à jour du logiciel sont possibles sans changement du dispositif matériel ce qui permet les versions correctives ou améliorations mineures sans contraintes supplémentaires. Cette flexibilité est un avantage considérable pour la plupart des éditeurs de logiciels.

4•11 Dispositif matériel fixe ou amovible

Selon ses applications, un éditeur de logiciel peut décider d'utiliser un dispositif matériel fixe ou amovible.

Si le dispositif matériel est fixe, c'est-à-dire installé de manière permanente dans une machine, le logiciel ne peut s'exécuter que dans cette machine. Par exemple, dans le cas d'un logiciel embarqué fonctionnant sur une machine dédiée qui ne doit pas être transporté sur une autre pour des questions de sécurité ou d'assurance qualité.

Si le dispositif matériel est amovible, celui-ci peut être déplacé d'une machine à une autre. Seule la machine ayant le dispositif matériel connecté peut exécuter le logiciel. Par exemple, l'utilisation au travail et au domicile du même programme protégé à condition de déplacer le dispositif matériel.

5 • DÉVELOPPEMENT DE PROGRAMME

5•1 Implications de l'utilisation de "Validy Technology" dans le développement

"Validy Technology" introduit un minimum de modifications lors de l'écriture ou de la protection d'un programme, le compilateur Validy se charge d'effectuer pratiquement toutes les modifications.

5•2 Protection contre le piratage, pas de librairie de sécurité

Pour la protection d'un programme contre le piratage, "Validy Technology" ne nécessite aucun appel de fonction appartenant à une librairie de sécurité. Les seules modifications consistent à choisir et à marquer les variables qui doivent résider dans le dispositif matériel.

Concrètement, il suffit d'insérer une simple ligne de commentaire devant la déclaration de chaque variable choisie. Le compilateur Validy se charge alors de générer les instructions de sécurité nécessaires pour la mise en œuvre des différentes facettes de "Validy Technology".

5•3 Processus de debug

Une fois les variables choisies et marquées, le développeur commence par compiler son programme avec son compilateur habituel et effectue le debug avec son debugger habituel.

Une fois satisfait du résultat, le développeur compile le programme avec le compilateur Validy et effectue le debug avec un debugger et un simulateur du dispositif matériel. Il vérifie de cette manière que le programme protégé fonctionne correctement.

Finalement, le développeur compile la version définitive de son programme avec le compilateur Validy et teste la version définitive avec un dispositif matériel Validy.

Remarque : Lors de la compilation finale, le renommage est effectué à l'aide d'un dispositif matériel de compilation : le développeur n'a donc pas accès à la fonction de renommage ce qui empêche toute fuite à ce niveau.

5•4 Protection de l'intégrité

L'essentiel de la protection de l'intégrité d'un programme s'effectue automatiquement par le compilateur Validy.

Par exemple, une vérification automatique du graphe d'appel est réalisée en rajoutant automatiquement les instructions de sécurité qui contrôlent l'arborescence du programme.

En cas de besoin de protection supplémentaire spécifique, une librairie est à la disposition du développeur.

5•5 Compilateur Validy

Afin d'avoir un maximum d'indépendance vis-à-vis des plate-formes supportées, Validy a développé un compilateur qui travaille au niveau byte-code. Ce compilateur Validy prend comme source du byte-code Java ou CIL (langage intermédiaire de la plate-forme .NET), ajoute la protection "Validy Technology" et régénère respectivement du byte-code Java ou CIL.

Ce compilateur permet de supporter les langages suivants : Java, C#, Visual Basic .Net, C++ .Net, ... sur toutes les machines supportées par ces langages.

5•6 Autres compilateurs

Validy envisage le développement d'une version de gcc ajoutant la protection "Validy Technology". Validy envisage aussi le développement d'un recompilateur binaire Pentium pour protéger un programme à partir de son code exécutable.

5•7 Outils additionnels

Validy développera les outils additionnels aidant à la mise en œuvre de la protection et notamment un outil de profiling pour l'aide au choix des variables ou pour le choix automatique des variables.



6 • IMPLICATIONS AU NIVEAU DÉPLOIEMENT



6•1 Les différents supports

La protection "Validy Technology" nécessite obligatoirement l'utilisation d'un dispositif matériel sécurisé qui peut prendre aujourd'hui différents aspects : clef USB, carte à puce, composant électronique, ...

Les clefs USB et les cartes à puce sont amovibles alors que les composants électroniques sont fixes (pour les avantages respectifs, voir paragraphe 4.1.1).

Les dispositifs matériels sécurisés sont fournis par Validy et sont appelés VSM (Validy Secure Module). Ils intègrent les derniers progrès en matière de sécurité et notamment les contre-mesures pour parer les attaques du type SPA, DPA, DEMA, SEMA, ...^(*)

6•2 La production

Les VSMs clients doivent être personnalisés à partir des informations ayant servi lors de la compilation et contenues dans le VSM maître (voir paragraphe 5.3). Cette personnalisation est totalement indépendante de Validy qui n'a aucun accès à ces informations.

La logistique et les moyens mis en œuvre pour la personnalisation peuvent varier selon le contexte de production :

Pour les petites quantités, les informations sont directement transférées du VSM maître vers les VSMs clients.

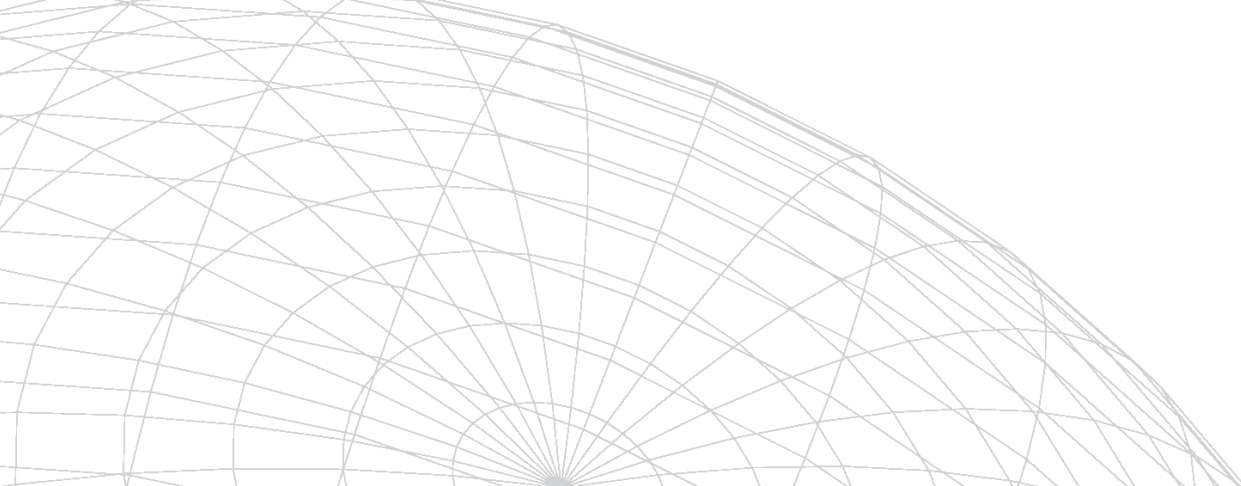
Pour les grosses quantités, les informations sont transférées du VSM maître vers des VSMs de production. Ces VSMs de production, situés directement sur les machines de production ou à distance, transmettent à leur tour les informations aux VSMs clients, localement ou via Internet.

La mise en œuvre de techniques standard de cryptographie permet le transfert sécurisé des informations, sans risque de fuite, entre les différents VSMs et ce quelle que soit la distance.

6•3 La maintenance

Le mode de maintenance est spécifique à chaque modèle de business, ce point est abordé dans les livres blancs modèles économiques.

^(*) : SPA : Simple Power Analysis, DPA : Differential Power Analysis, DEMA : Differential ElectroMagnetic Analysis, SEMA: Simple ElectroMagnetic Analysis



SA Validy
ZI - 5, rue Jean Charcot
26100 Romans - France
info@validy.com



Validy Net Inc.
1001 SW Fifth Avenue
suite 1100
Portland, OR 97204 - USA