Accès sécurisé aux ressources de test IEEE 1687 et aux crypto-processeurs légers dans le contexte des IoT.

Vincent Reynaud

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : **NANO ELECTRONIQUE ET NANO TECHNOLOGIES**

Arrêté ministériel : 25 mai 2016

Présentée par

Vincent Reynaud

Thèse dirigée par **Régis LEVEUGLE**, Professeur, Université Grenoble Alpes
Et co-encadrée par **Paolo MAISTRI**, Chargé de Recherche, CNRS

préparée au sein du **Laboratoire Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)**

Accès sécurisé aux ressources de test IEEE 1687 dans le contexte IoT.

Secured access to IEEE 1687 test resources in the IoT context.

Thèse soutenue publiquement le **9 mars 2021**,

devant le jury composé de :

Monsieur, Giorgio, DI NATALE,
Directeur de laboratoire, Université Grenoble Alpes, Président
Monsieur, Jean-Max, DUTERTRE,
Professeur, Ecole Nationale Supérieure des Mines de Saint-Etienne, Rapporteur
Monsieur, Alberto, BOSIO,
Professeur, Ecole Centrale Lyon, Rapporteur
Monsieur, Lionel, TORRES,
Professeur, Université de Montpellier, Examinateur
Monsieur, Régis, LEVEUGLE,
Professeur, Université Grenoble Alpes, Directeur de thèse
Monsieur, Paolo, MAISTRI,
Chargé de Recherche, CNRS, Co-encadrant de thèse

Table des matières

Liste des figures 5

Glossaire 8

Introduction générale 10

Chapitre I. Etat de l'art : test, attaques du matériel et contremesures 12

I.1. Test des circuits 12

I.1.1. Généralités 12

I.1.2. Chaîne de scan 12

I.1.3. Boundary scan 13

I.1.4. Réseau de Scan Reconfigurable 14

I.1.5. Equipements de test 16

I.2. Menaces de sécurité et solutions hors chaînes de scan 17

I.2.1. Attaques non invasives 17

I.2.2. Attaques invasives ou semi-invasives 18

I.2.3. Défenses possibles 19

I.3. Menaces de sécurité dues aux chaînes de scan 20

I.3.1. Protection des chaînes de scan 21

I.3.2. Protections inhérentes 21

I.3.3. Protection des données 22

I.3.4. Gestion d'accès 23

I.3.5. Chiffrement des chaînes de scan 26

I.4. Conclusion 27

Chapitre II. Conception d'un système de sécurisation d'infrastructure de test 29

II.1. Projet HADES 29

II.2. Synthèse des méthodes de gestion d'accès 30

II.3. Politique de gestion de clé 34

II.3.1. Configurations et clés de configuration 34

II.3.2. Génération procédurale de clé 35
II.3.3. Clés reconfigurables 36
II.3.4. Distribution 36
II.3.5. Identification d'utilisateur 37
II.4. Architecture du système 38
II.4.1. Contrôleur et périphériques 38
II.4.2. Contrôleur d'authentification 40
II.5. Protocole et machine à états 41
II.5.1. Protocole d'authentification 41
II.5.2. Machine à états du protocole 42
II.6. Chiffrement des vecteurs 44
II.6.1. Protocole d'autorisation et de chiffrement 44
II.6.2. Architecture du système 45
II.7. Conclusion 47
Chapitre III. Test et chiffrement dynamique 48
III.1. Automatisation de l'authentification 48
III.1.1. Equipement de test et authentification 48
III.1.2. Génération dynamique en ligne des vecteurs de test. 49
III.1.3. Utilisation de MAST pour l'authentification SSAK 51
III.2. Chiffrement interne 56
III.2.1. Confidentialité des données hors de leurs segments 56
III.2.2. Chiffrement par SIB 57
III.2.3. Gestion des déphasages de clés et contrôle 58
III.3. Conclusion 61
Chapitre IV. Démonstrateurs et Résultats 62
IV.1. Démonstrateurs 62
IV.1.1. Authentification seule 62

IV.1.2. Authentification et chiffrement 63
IV.1.3. Authentification et MAST 68
IV.1.4. Chiffrement interne 70
IV.2. Caractéristiques des cryptoprocesseurs 75
IV.2.1. Liste d'étude 76
IV.2.2. Coûts d'intégration 80
IV.2.3. Performances 81
IV.2.4. Compromis 83
IV.3. Etude d'impact 83
IV.3.1. Méthodologie 84
IV.3.2. Flot d'évaluation 84
IV.3.3. Résultats 86
Conclusion générale 92
Références 94
Publications 96

Liste des figures

Figure I.1 Exemple de chaîne de scan 13
Figure I.2 Architecture de test d'un circuit imprimé 14
Figure I.3 Test Data Register 14
Figure I.4 Segment Insertion Bit. 15
Figure I.5 Infrastructure de test sur SOC 16
Figure I.6 Environnement de test 17
Figure I.7 Interface IJTAG des registres de tour d'un chiffreur AES. 20
Figure I.8 Chaînes de scan multiples avec compacteur 22
Figure I.9 Chaîne de scan avec brouillage 23
Figure I.10 Schéma contextuel du LSIB 24
Figure I.11 Schéma du S²IB 25
Figure I.12 Contrôle des S²IB 25
Figure I.13 Protocole d'authentification du contrôleur de S²IB 26
Figure I.14 Schéma de principe du chiffrement de chaîne de scan 27
Figure II.1 Surface d'implémentation supplémentaire de LSIB et FGA sur le circuit de test T512505 31
Figure II.2 Temps d'ouverture de tous les SIB en fonction du nombre de SIB protégées 31
Figure II.3 Représentation des topologies en ligne et en colonne 32
Figure II.4 Temps d'ouverture des SIB avec une topologie en ligne 32
Figure II.5 Temps d'ouverture des SIB avec une topologie en colonne 33
Figure II.6 Vecteur de numérotation et ensemble de segment 35
Figure II.7 Génération procédurale de SSAK (a) Avec AES (b) avec SHA256 36
Figure II.8 Distribution des clés et des identifiants 37
Figure II.9 Clé de configuration personnelle 38
Figure II.10 Contrôleur et ses périphériques 39
Figure II.11 Contrôleur SSAK et RSN 40
Figure II.12 Implémentation RTL du contrôleur 41
Figure II.13 Protocole d'authentification 42
Figure II.14 Machine à états du contrôleur SSAK 43
Figure II.15 Chiffrement de chaîne de scan et autorisation SSAK 45
Figure II.16 Unité de chiffrement 46

Figure III.1 Génération des vecteurs de test statiques 49
Figure III.2 Environnement de test MAST 50
Figure III.3 Portabilité de MAST 51
Figure III.4 Authentification SSAK automatisée par le système MAST 52
Figure III.5 Diagramme de classe du plugin SSAK 53
Figure III.6 Utilisation de différents cryptoprocesseurs dans MAST 54
Figure III.7 Exemple d'architecture incluant MAST et SSAK 54
Figure III.8 Authentification par PDL (a), authentification par MAST (b), authentification partielle avec MAST (c). Seuls PDL 1 et 2 requièrent une authentification. 55
Figure III.9 Modèles d'attaquant : (a) externe, (b) interne 56
Figure III.10 Approche théorique de la protection des données de segments à l'intérieur du circuit 57
Figure III.11 eSIB : Ajout d'une fonction de chiffrement au S²IB 58
Figure III.12 Propagation de la clé de flux dans la chaîne de scan sécurisée .. 58
Figure III.13 Schéma du cas général du chiffrement intene pour l'écriture et la lecture 59
Figure III.14 Chaîne de scan interne avec plusieurs chiffrement internes 60
Figure IV.1 Architecture générale du premier démonstrateur 63
Figure IV.2 Architecture du démonstrateur SSAK et chiffrements fusionnés sur carte JTAG 64
Figure IV.3 Code couleur des LED du démonstrateur authentification et chiffrement 65
Figure IV.4 Version du démonstrateur SSAK et chiffrement juxtaposés (a), seulement SSAK (b), seulement le chiffrement (c) et version sans protection (d) 66
Figure IV.5 Répartition de l'utilisation des LUT dans les solutions juxtaposées et fusionnée 68
Figure IV.6 Démonstrateur MAST et SSAK 69
Figure IV.7 Démonstrateur de chiffrement interne 71
Figure IV.8 Diagramme UML de la gestion de model de chiffrement interne ... 72
Figure IV.9 Pseudo code des méthodes updatePos 73
Figure IV.10 Pseudo codes des méthodes addEncryptionPoint 75
Figure IV.11 Schéma de fonctionnement d'un tour de SHA-2 77
Figure IV.12 Structure en éponge 78

Figure IV.13 Chiffrement de flux avec Trivium 79
Figure IV.14 Fonction de permutation du hacheur Quark..... 79
Figure IV.15 Schéma de principe de GRAIN..... 80
Figure IV.16 Nombre de portes NAND équivalentes utilisées pour chaque cryptoprocesseur 81
Figure IV.17 Débit binaire des cryptoprocesseurs..... 82
Figure IV.18 Carte d'efficacité vitesse/coût/sécurité des cryptoprocesseurs.... 83
Figure IV.19 Flot de synthèse et de simulation 84
Figure IV.20 Fichier de circuit de test configurable 86
Figure IV.21 Temps d'accès aux segments du circuit T512505 en fonction du nombre de segments protégés avec un hacheur SHA2 87
Figure IV.22 Temps d'accès aux segments du circuit T512505 en fonction du nombre de segments protégés en utilisant le QUARK LENT 88
Figure IV.23 Coût d'implémentation des différentes approches en utilisant l'AES128 sur le circuit de stimulation T512505..... 89
Figure IV.24 Coût d'implémentation des différentes approches en utilisant le GRAIN sur le circuit de stimulation T512505 89
Figure IV.25 Répartition des coûts pour chaque stratégie pour protéger 100 segments dans le circuit t512505 90

Glossaire

AES	Advanced Encryption Standard
ATE	Automated Test Equipement
BISR	Built-In Self Repair
BIST	Built-In Self-Test
BSR	Boundary Scan Register
CRC	Controle de Redondance Cyclique
cSIB	Configurable Segment Insertion Bit
CUT	Circuit Under Test
DES	Data Encryption Standard
DFT	Design For Test
DUT	Design Under Test
EEPROM	Electrically-erasable programmable read-only memory
eSIB	encryption Segment Insertion Bit
FF	Flip Flop
FGA	Fine Grained Access
FPGA	field-programmable gate array
FSM	Finite State Machine
HCERES	Haut Conseil de l'évaluation de la recherche et de l'enseignement supérieur
ICL	Instrument Connectivity Language
IoT	Internet of Things
IVSW	International Vérification and Security Workshop
JTAG	Joint Test Action Group
LFSR	Linear Feedback Register
LSB	Least significant bit
LUT	Look Up Table
MAE	Machine à Etats
MAST	Manager for System-Centric Test
NFSR	Nonlinear Feedback Register
NSA	National Security Agency
P-box	Table de permutation
PCB	Printed Circuit Board
PDL	Procedure Description Language
RAZ	Remise à zéro
ROM	Read Only Memory
RSN	Reconfigurable Scan Network
RTL	Regsiter Transfert Level
S^2IB	Secure Segment Insertion Bit
S-box	Table de substitution
SIB	Bit d'Insertion de Segment (Segment Insertion Bit)
SNR	Rapport Signal sur Bruit
SoC	System on Chip
SSAK	Segment Set Authorization Key
TAP	Test Access Port

TDR	Test Data Register
test flow	Environnement de test
TRNG	True Random Number Generator
VHDL	Very High-Speed Integrated Circuit Hardware Description Language

Introduction générale

Chaque année, des milliards de puces électroniques sont fabriquées et vendues à travers le monde ; il peut s'agir de puces d'identification comme d'un processeur de dernière génération ultra puissant. Quelle que soit leur complexité ils doivent fonctionner après leur fabrication et durant leur durée de vie prévue. Pour s'en assurer tous les exemplaires sont méticuleusement testés et la plus grande partie possible des transistors ou connexions est pris en compte. Cela représente une quantité de travail non réalisable sans une stratégie mise en place. Pour permettre la réalisation de ces tests aussi exhaustifs que possible dans la durée impartie, les concepteurs doivent intégrer au circuit des éléments prévus pour faciliter le test. C'est ce qui s'appelle la conception pour le test.

Pour résoudre l'une des principales difficultés, à savoir le test de circuits séquentiels, des infrastructures permettant d'accéder à la plupart des registres et éléments mémorisant du circuit son utilisés. Elles permettent de faciliter l'observation et le contrôle des différents éléments du circuit.

Si cette facilité de contrôle et d'observation sert le travail des testeurs, ils ne sont pas les seuls à l'utiliser. En effet, les infrastructures de test peuvent également être utilisées pour effectuer du débogage de logiciels en permettant d'accéder par exemple aux registres des processeurs, ou utilisés pour programmer des composant logiques configurables ou certains microcontrôleurs. Malheureusement, ces réseaux sont aussi susceptibles de servir de moyen d'accès facile pour d'éventuels attaquants du circuit. En effet la contrôlabilité et l'observabilité offertes par cet outil peuvent permettre à des personnes mal intentionnées de récupérer des données sensibles ou d'induire un comportement non prévu.

Pour lutter contre ce risque, plusieurs techniques ont été développées pour éviter que les infrastructures de test restent une faille de sécurité importante. Il peut s'agir par exemple de la destruction des signaux d'accès après le test de fin de fabrication, de chiffrement de données de scan ou de l'accès avec autorisation de connexion. Le Chapitre I détaille l'état de l'art actuel sur ces infrastructures de test, leur utilisation et leur protection.

L'objectif de cette thèse est de proposer une solution innovante – appelée SSAK – capable de ne permettre l'accès à des parties déterminées du circuit qu'aux seuls utilisateurs autorisés à y accéder. La solution proposée permet également d'assurer la confidentialité des données de test à l'extérieur du circuit au moyen d'une technique de chiffrement issue de l'état de l'art. Les détails du développement de cette solution sont présentés dans le Chapitre II.

Le test des circuits, comme les efforts de conception pour faciliter ce test, est fortement contraint par le temps. Il doit être effectué le plus rapidement possible sous peine de ralentir la commercialisation d'un produit et d'augmenter fortement les coûts. Les nouveautés introduites dans l'environnement, susceptibles d'introduire du travail

d’adaptation chronophage, peuvent être perçues comme risquées. Si un protocole innovant veut pouvoir être utilisé il faut qu'il soit le plus possible compatible avec les équipements de test utilisés. Ces équipements sont la plupart du temps incapables de faire du test interactif et incapables de pouvoir réagir en ligne aux données issues des infrastructures de test, ce qui rend compliqué l'utilisation d’une solution d’authentification sécurisée. Pour remédier à ce problème l'outil MAST, développé dans le laboratoire, est utilisé. Grâce à sa capacité à interagir dynamiquement avec l'infrastructure de test, il est possible d'intégrer l’authentification SSAK aux procédures de te d’une façon automatique et transparente pour le testeur.

Pour étendre la solution qui permet de garantir la confidentialité des données, l'infrastructure d'autorisation d'accès est réutilisée pour offrir un chiffrement aux bornes de chaque partie critique de la chaîne de scan, empêchant ainsi à d'éventuels espions placés à l'intérieur du circuit de récupérer des données de test. La conception du test automatisé et l'approche permettant de garantir la confidentialité interne seront présentées dans le Chapitre III.

Le Chapitre IV est dédié à la preuve de faisabilité des solutions, ainsi qu'à l'étude de leur impact en termes de coût supplémentaire impliqué et de temps de test du circuit. Les preuves de faisabilité sont effectuées grâce à des démonstrateurs implémentés sur des circuits de logique programmable (FPGA) ou grâce à une simulation. Pour les études d'impact, les solutions sont mises en œuvre sur des circuits de test de référence et comparées aux solutions issues de la littérature.

Chapitre I.
Etat de l'art : test, attaques du matériel et contremesures

Ce premier chapitre a pour but de présenter certains éléments essentiels dans le domaine du test des circuits ainsi que les techniques liées à leur attaque et à leur protection.

Les deux parties du champ du test numérique qui nous intéressent dans ce chapitre sont la conception pour le test (DFT) et la pratique du test des circuits dans l'industrie. Le sujet sera principalement abordé sous l'aspect de la conception, qui est l'aspect du test le plus utilisé par les chapitres suivants, néanmoins des bases sur les outils et le déroulement des tests seront présentées succinctement.

La suite du chapitre est consacrée à l'état de l'art sur la sécurité matérielle, tout d'abord en abordant les attaques qui n'exploitent pas les infrastructures de test et leurs contre-mesures. L'emphase sera ensuite mise sur les attaques exploitant les failles de ces infrastructures. Enfin une section détaillera les principales options déjà publiées pour protéger les infrastructures de test.

I.1. Test des circuits

I.1.1. Généralités

En micro-électronique, le processus d'implémentation physique des circuits n'aboutit pas exactement au même résultat pour tous les exemplaires, certaines variations apparaissent faisant différer leurs performances, allant dans les pires des cas à les rendre inopérants. Le test est une discipline dont le but est de vérifier que chacun des exemplaires produits ait un comportement cohérent avec celui attendu. Conséquemment à la complexification des circuits au fil des années, les tests devinrent plus difficiles, ce qui poussa l'industrie à développer des stratégies pour faciliter ces tests.

Afin de mener des tests efficaces, les notions de contrôlabilité et d'observabilité sont fondamentales : la contrôlabilité est la capacité de forcer le système dans un état précis, et l'observabilité est la capacité de connaitre l'état du circuit. Pour vérifier efficacement le bon fonctionnement d'un circuit il faut avoir une ample maîtrise de ces deux capacités. Dans cet objectif, les concepteurs vont ajouter du matériel dans les circuits pour les rendre plus facilement testables, augmentant ainsi ces deux paramètres, pratique nommée conception pour le test ou *Design For Test* (DfT) en anglais.

I.1.2. Chaîne de scan

Suivant cette démarche, les chaînes de scan sont des outils de test permettant de lire et d'écrire facilement dans les bascules des circuits séquentiels. Cette chaîne

est créée en mettant bout à bout les bascules de manière à former un registre à décalage. Dans le circuit d'illustration de la Figure I.1, les éléments rajoutés pour former la chaîne de scan sont représentés en bleu. Le circuit possède alors deux modes de fonctionnement. Lorsque le signal *shift* est activé les bascules sont mises en série et elles sont en mode test. Autrement, le circuit est en mode de fonctionnement normal. Une opération de test se fait en trois étapes : l'étape *scan in* permet en mode test d'insérer les données de test aussi appelées vecteurs. L'étape s*can-capture,* qui en mode normal, permet de stimuler le circuit combinatoire et de mémoriser les réponses dans les bascules. Enfin, l'étape *scan-out* en mode test permet de les récupérer.

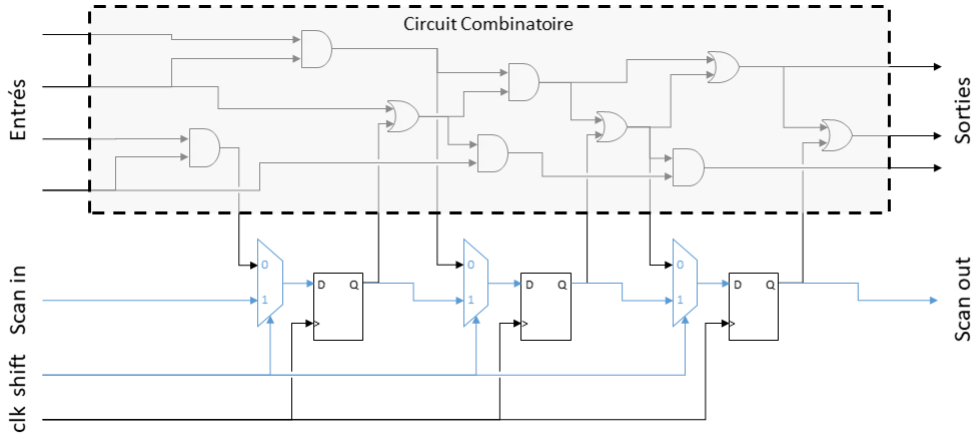


Figure I.1 Exemple de chaîne de scan

L'un des problèmes qui apparaissent lors de l'augmentation de la taille et de la densité des circuits est l'élévation de température due aux grandes puissances consommées sur des petites surfaces lors des tests. Cela peut mener d'une diminution des performances à une destruction des circuits. Plusieurs méthodes ont émergé pour éviter ces conséquences, notamment l'utilisation de vecteurs de test présentant moins de transitions [1], ou la division de la chaîne en segments [2]. La division en segments de test permet de ne pas avoir à effectuer les tests simultanément sur tout le circuit et ainsi réduire la puissance instantanée consommée.

I.1.3. Boundary scan

Dans le but de faciliter la vérification et le test des circuits imprimés (PCB), le *Joint Test Action Group* (JTAG) développa une proposition qui devint ensuite la norme IEEE 1149.1 [3] qui introduit une chaîne de scan nommée *Boundary Scan Register* (BSR) et un contrôleur de test. Ils sont tous les deux accessibles depuis l'extérieur du circuit grâce au *Test Access Port* (TAP). Comme représenté sur la Figure I.2, la chaîne parcoure dans différents circuits les cellules reliées aux broches d'entrée et de sortie. Grâce à ce dispositif il est possible d'effectuer des tests de continuité, qui consistent à vérifier l'intégrité des interconnexions entre les circuits intégrés ainsi que l'absence de court-circuit. Il est en outre possible d'isoler les

circuits pour pouvoir tester leurs fonctionnalités indépendamment, en exploitant par exemple les chaînes de scan internes.

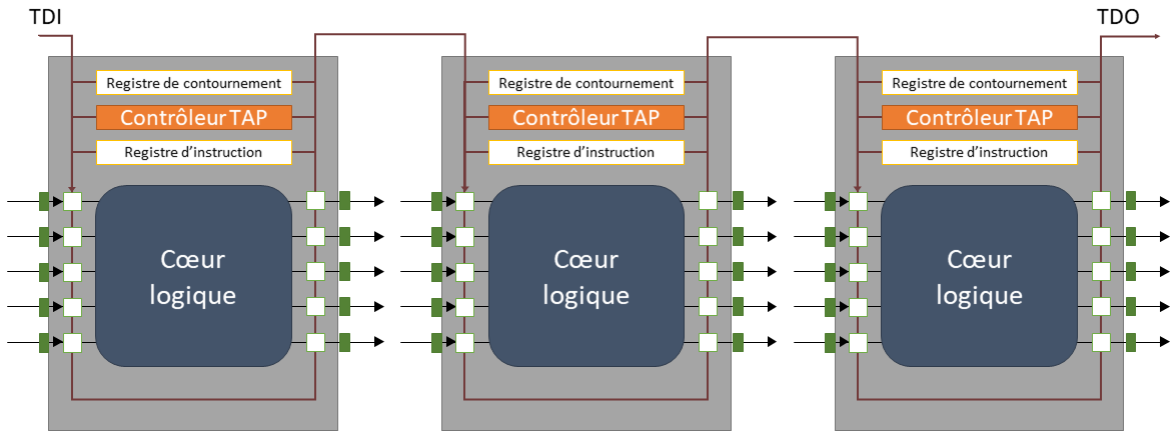


Figure I.2 Architecture de test d'un circuit imprimé

La chaîne de scan possède quatre signaux de contrôle : *select* qui active ou désactive le mode test ; *shiftEn* qui permet de passer la chaîne en mode registre à décalage ; *captureEn qui* est utilisé pour charger des données dans la chaîne et *updateEn qui* est utilisé pour transmettre les données. La Figure I.3 représente un registre de donnée de test (TDR : *Test Data Register*) de trois bits. Ces registres sont notamment utilisés pour contrôler des instruments.

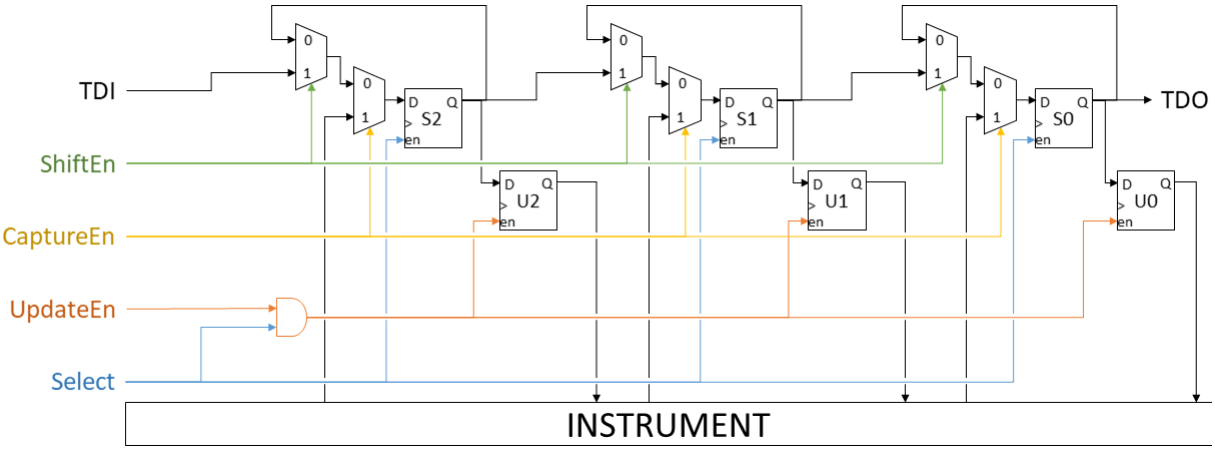


Figure I.3 Test Data Register

I.1.4. Réseau de Scan Reconfigurable

L'utilisation d'une unique chaîne de scan pour tester des circuits dont la taille et la complexité croissent continuellement, devint rapidement inadaptée. En effet ces circuits requerraient une très longue chaîne de scan, rendant l'accès aux bascules chronophage. Les chaînes de scan ont tout d'abord été divisées en plusieurs chaînes parallèles, en établissant un compromis entre le temps de test, le nombre de broches de commande ajoutées pour le test et la consommation induite

par les tests exécutés en parallèle. Afin de réduire la quantité de données transmise pour le test, des mécanismes de compression/décompression ont été rajoutés. Les durées de test ont cependant continué à augmenter avec la complexité des circuits. De plus, le test a dû prendre en compte un nombre croissant de paramètres, liés par exemple aux variabilités.

L'utilisation d'une chaîne de scan ou même de plusieurs chaînes parallèles est fastidieuse pour des grands circuits. Par ailleurs, le test de fin de fabrication qui vise l'intégralité du circuit n'est pas le seul important. En effet, les structures de test sont également utilisées pendant la durée de vie du circuit pour vérifier l'état et le fonctionnement des différentes parties. Des ajouts ont donc été développés pour rendre la structure de test flexible et adaptable à différents scénarii de test. Le standard IEEE 1687-2014 [4] fut alors créé. L'un des ajouts est l'utilisation d'instruments de test intégrés incluant les thermomètres et autres sondes ainsi que des systèmes autotest intégrés "*Built-In Self-Test*" (BIST).

Pour rendre l'infrastructure plus flexible, il a été ajouté la possibilité de reconfigurer dynamiquement les chaînes de scan lors des tests. Pour cela, un des éléments introduits par le standard est le Bit d'Insertion de Segment (SIB : Segment Insertion Bit), illustré par la Figure I.4. Cet élément est un multiplexeur de chaînes de scan autocontrôlé ; le registre U contrôle l'état courant du SIB et se pilote comme le registre de donnée de test vu précédemment.

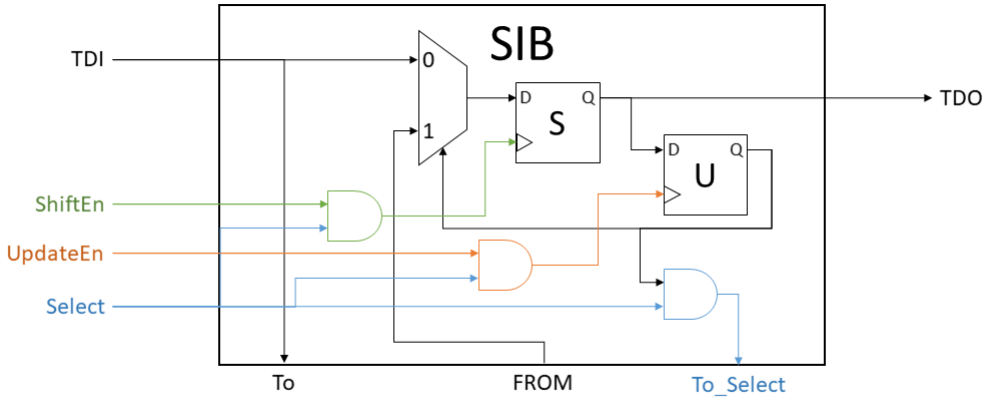


Figure I.4 Segment Insertion Bit.

Grâce aux SIB, il est possible de créer une infrastructure de test reconfigurable avec plusieurs niveaux hiérarchiques, appelée en anglais *Reconfigurable Scan Network* (RSN) comme l'illustre la Figure I.5. Dans cet exemple, le circuit est divisé en plusieurs blocs, pour chacun d'entre eux la chaîne interne peut être accédée en activant un SIB. L'intérieur du bloc CPU est détaillé, il montre ainsi que les accès aux sous-éléments peuvent également se faire par l'activation d'un SIB. Le sous-bloc des caches montre le dernier niveau hiérarchique de cette figure, mais il n'existe pas de limite théorique au nombre de niveaux. Cette segmentation de la chaîne de scan permet aux utilisateurs qui ont besoin d'effectuer des tests partiels

ou de n'utiliser qu'une partie réduite de la chaîne de scan, de le faire sans avoir à émettre ou lire des vecteurs de test qui couvrent l'intégralité de la chaîne de scan.

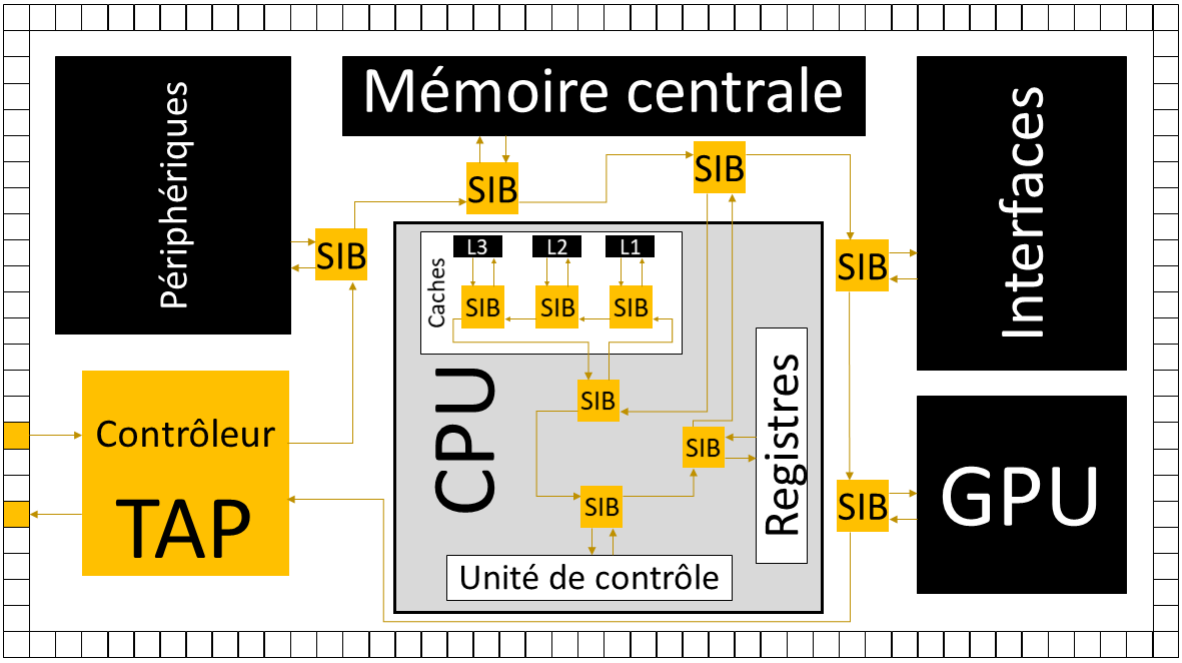


Figure I.5 Infrastructure de test sur SOC

Parmi les différentes utilisations des infrastructures en dehors du test de fin de fabrication, peut être cité le débogage logiciel, qui permet d'accéder aux registres, aux caches et autres éléments du processeur. Les unités de mémoires sont régulièrement testées et les cellules endommagées peuvent être remplacées, grâce à des unités d'autotest et d'autoréparation ou *built-in self repair* (BISR) [5]. Certains firmware sont programmés dans les puces en utilisant les réseaux de test, c'est le cas par exemple des réseaux de portes programmables in situ (FPGA).

I.1.5. Equipements de test

Le test en fin de fabrication des circuits électroniques est réalisé selon un environnement bien défini décrit par la Figure I.6 issue de [6]. Des outils regroupent les informations venant des étapes de conception, c'est-à-dire la description niveau portes (*netlist*) du circuit plus les informations relatives à la DFT comme l'architecture du RSN décrite par l'*Instrument Connectivity Language* (ICL) et les instructions de test au format *Procedure Description Language* (PDL) en ce qui concerne le standard IEEE1687. Grâce à ces informations, les vecteurs de test peuvent être générés et utilisés par des équipements de test automatisés (ATE), pour éprouver des instances physiques du circuit.

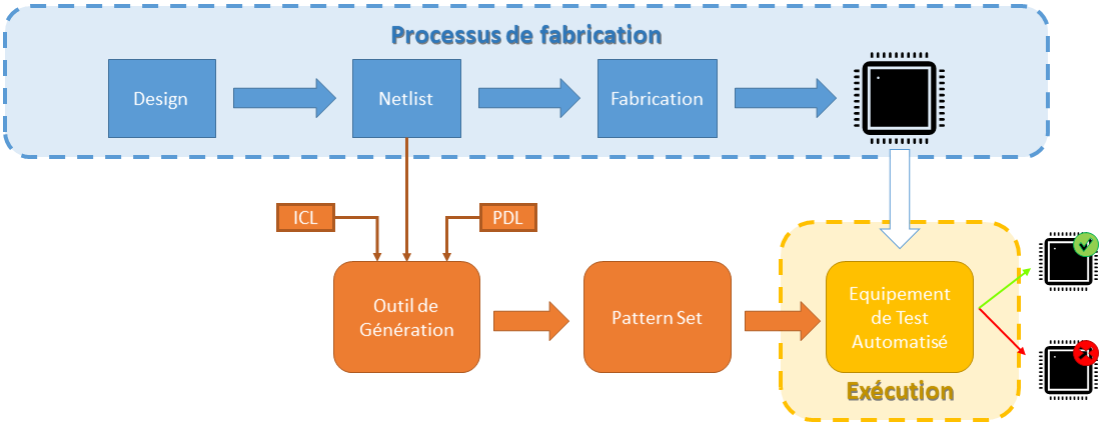


Figure I.6 Environnement de test

I.2. Menaces de sécurité et solutions hors chaînes de scan

Les circuits intégrés recèlent parfois des données ou d'autres éléments convoités par des personnes malveillantes. Il peut s'agir de données bancaires, de clés de chiffrement, ou bien de l'architecture même du circuit. Bien que l'utilisation normale du circuit ne permette pas d'accéder à ces informations, certains utilisateurs n'hésitent pas à lancer des attaques contre les circuits. Etant donnée la nature théoriquement inviolable des algorithmes et protocoles de sécurité, les attaquants peuvent se tourner vers le matériel pour trouver des failles dans l'implémentation. Cette section va résumer différentes techniques d'attaque sur le matériel en dehors de celles utilisant la chaîne de scan qui elles feront l'objet d'une partie dédiée. Ces attaques se divisent en deux catégories : les attaques non invasives et invasives ou semi-invasives. Ce classement révèle l'impact qu'ont ces techniques sur le circuit : tandis que les attaques non invasives laisseront la puce intacte, les attaques invasives peuvent la détériorer, voire la rendre complètement inutilisable. Cette partie présente également des contremesures à ces attaques, parmi celles présentes dans la littérature.

I.2.1. Attaques non invasives

Les attaques par canaux auxiliaires sont très répandues car simples à mettre en place, peu onéreuses et laissant le circuit intact. Il s'agit d'observer les traces que laisse le circuit dans son environnement : la durée de calcul, la consommation d'énergie, ou le rayonnement électromagnétique, voire acoustique [7].

Dès la fin des années 90, des techniques d'attaque mettant en œuvre les canaux auxiliaires ont apparu. Le cryptoprocesseur standard de l'époque, basé sur le *Data Encryption Standard* (DES), en fut par exemple une victime [8]. L'attaque subie se base sur la puissance consommée par le circuit. La trace mesurée lors d'une opération de chiffrement permet d'identifier en outre des cycles d'horloge et les différentes étapes de l'algorithme. L'enchaînement de certains cycles du DES se fait grâce à des branchements conditionnés sur les données ; il est alors possible,

en observant la courbe de consommation résultant des enchaînements de cycles, de remonter jusqu'aux données. Les caractéristiques asymétriques de la consommation en fonction des données employées permettent également des analyses statistiques (analyses différentielles DPA, CPA, etc.).

Une autre méthode d'attaque par canal auxiliaire dont l'utilisation a fortement augmenté est l'utilisation de l'empreinte électromagnétique d'un circuit. Il peut par exemple être cité plus récemment l'attaque sur le chiffreur *Advanced Encryption Standard* (AES)[9], [10] ou sur des chiffreurs plus légers comme TWINE [11]. Ces derniers, comme beaucoup d'autres chiffreurs, utilisent plusieurs tours de chiffrement, ainsi que plusieurs sous-clés de chiffrement. Le protocole de l'attaque consiste à effectuer de nombreux chiffrements et à sauvegarder les résultats sous forme de couples cryptogramme / courbe électromagnétique. Avec de nombreux résultats ainsi relevés les attaquants sont capables de faire une étude statistique. Avec ce type d'attaque, il est possible d'obtenir toutes les clés de tour nettement plus rapidement que par la force brute.

Ces méthodes se sont démocratisées, car peu chères et ne nécessitant pas beaucoup de matériel. Elles ont beaucoup évolué et permettent aujourd'hui d'attaquer avec succès des circuits pourtant conçus avec des protections sophistiquées. En outre il est difficile de détecter de telles attaques puisqu'elles se basent exclusivement sur l'observation du système.

I.2.2. Attaques invasives ou semi-invasives

A l'inverse des attaques précédentes, les attaques invasives ont un impact sur l'état du circuit à l'issue de l'attaque, certaines méthodes pouvant même détruire complètement une puce. Par exemple, le travail publié en 2017 dans [12] implémente deux attaques. Elles sont utilisées pour récupérer les données d'une mémoire morte (ROM). La première consiste en une observation au microscope optique sur la mémoire de deux circuits sécurisés des années 2000 en technologie CMOS 0.35 µm. Pour être capable de voir les couches de métal du circuit contenant les données des mémoires il a été nécessaire d'ôter le boitier des puces à l'aide d'une solution acide. Il est alors possible de procéder à l'observation de la mémoire et ainsi récupérer visuellement les données de la mémoire en analysant les connexions des couches de métal. Cependant, ces données se révèlent être chiffrées ; pour les décrypter les auteurs utilisent alors une attaque par microsondes. Elles sont branchées de part et d'autre du bus de communication qui relie le processeur et la mémoire dans laquelle se trouve le module de déchiffrage. En observant les échanges, les attaquants sont capables de créer une table de correspondance à l'aide de laquelle ils déchiffrent les données précédemment récupérées.

Les méthodes d'attaque invasives sont très coûteuses à cause du matériel qu'il faut pouvoir déployer. Ces méthodes détruisant souvent les circuits, elles ne peuvent pas être utilisées pour récupérer des informations utiles à un seul circuit.

En revanche, elles sont très efficaces pour faire de la rétro-ingénierie, ou comme dans l'exemple précédent pour récupérer les données d'une mémoire morte qui est commune à toutes les instances du circuit.

Parmi les attaques les plus dangereuses de nos jours, les attaques par laser permettent de créer des erreurs ciblées en cours de fonctionnement, afin d'en déduire les données manipulées par le circuit lors des calculs. Ces attaques dites "par fautes" se sont développées sur tous les algorithmes de chiffrement. Cela nécessite en général avec les technologies actuelles des attaques "semi-invasives" c'est-à-dire que les parties actives du circuit ne sont pas modifiées, mais le circuit est sorti de son boîtier et en général son substrat est aminci. Il y a donc un risque de détérioration du circuit pendant la préparation de l'attaque. D'autres méthodes de création d'erreurs existent, plus ou moins invasives. Par exemple, les attaques par injection électromagnétique permettent également de créer des erreurs ciblées en cours de fonctionnement, potentiellement de manière non invasive ou seulement en ouvrant le boîtier du circuit.

I.2.3. Défenses possibles

Pour se protéger des attaques par canal auxiliaire plusieurs stratégies ont été élaborées. Il est notamment possible d'utiliser des techniques de "randomisation" pour rendre les données fuitant dans l'environnement plus difficiles à analyser. Cette technique est par exemple utilisée pour protéger la clé privée d'un chiffreur RSA dans [13] en divisant une opération d'exposant en plusieurs sous-exposants dont la répartition est choisie aléatoirement. Il est également proposé d'utiliser des techniques de masquage [14] dont le principe est d'effectuer les opérations avec des données différentes de celles que l'on veut protéger, car modifiées par un masque aléatoire. Il est possible de citer d'autres techniques comme l'insertion de bruit pour diminuer le rapport signal sur bruit (SNR) des fuites de canal auxiliaire [15], ou le regroupement des instructions spécifiques d'algorithmes de chiffrement en blocs insécables dont les fuites ne donnent pas d'informations sur les données [16]. Des techniques purement matérielles ont aussi été proposées pour rendre la consommation et les émissions davantage indépendantes des données ; elles sont souvent basées sur des approches de type "double rail" ou logique complémentaire.

Certaines techniques sont aussi apparues pour protéger les circuits des attaques invasives, comme des détecteurs capacitifs [17] qui peuvent détecter des attaques par microsonde ou des boucliers de sécurité [18] qui forment un obstacle visuel à l'attaque par observation et un obstacle physique à l'attaque par sonde. Diverses techniques sont également employées pour lutter contre les attaques par fautes, souvent basées sur de la redondance (matérielle, temporelle ou d'information).

Cette partie n'est qu'un aperçu rapide des nombreuses approches qui ont été proposées. Les approches visant les attaques par chaînes de scan, davantage liées au travail de la thèse, seront plus détaillées ci-après.

I.3. Menaces de sécurité dues aux chaînes de scan

Pour augmenter leurs chances de réussite, les attaquants peuvent aussi utiliser les mécanismes introduits dans les circuits pendant leur conception, et notamment les différents types de chaînes de scan. A ceci, peuvent s'ajouter des menaces liées à des éléments introduits illégalement dans le circuit, couramment appelés les chevaux de Troie, mais ceci sort du cadre de cette thèse.

Les attaques basées sur les chaînes de scan sont non-invasives et peuvent permettre à l'attaquant, sans ouvrir le boîtier, d'avoir un contrôle et une observabilité avancés sur le circuit. S'ils ne sont pas protégés, ces accès forment une faille de sécurité majeure pour un circuit [19], que ce soit une simple chaîne de scan, un accès par *boundary scan* ou une chaîne de scan reconfigurable.

Les cryptoprocesseurs sont en particulier fréquemment la cible d'attaques par chaîne de scan, le but étant de récupérer la clé de chiffrement. Des attaques ont été réalisées sur des chiffreurs symétriques [21][22][23][24], sur des chiffreurs asymétriques [25] et des chiffreurs de flux [26]. La première étape des attaques est de trouver où se situent les registres des cibles sur la ou les chaînes de scan d'un circuit. Comme l'illustre la Figure I.7, les registres peuvent être placés de façon simple et linéaire (a) ou de façon plus chaotique, voire sur plusieurs chaînes de scan différentes (b,). S. S. Ali et al exposent des solutions pour retrouver les emplacements [24].

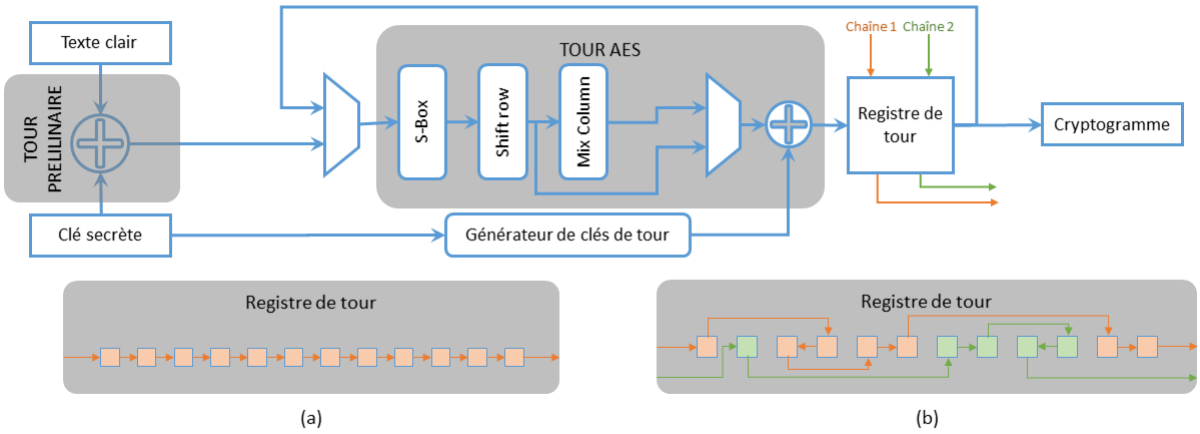


Figure I.7 Interface IJTAG des registres de tour d'un chiffreur AES.

Une fois la position des registres connue, il est possible de lancer les attaques en suivant trois étapes [22]. La première étape est d'appliquer un texte clair choisi. La seconde est de faire passer pendant un cycle d'horloge le cryptoprocesseur en mode fonctionnel. Enfin, la dernière étape est de récupérer le résultat intermédiaire dans le registre de tour grâce à la chaîne de scan. Ces trois étapes permettent d'obtenir des couples textes clairs / résultats intermédiaires. Le travail en différentiel permet de faciliter l'extraction de la clé car cela neutralise l'impact de la clé de tour et divise le tour en seize chemins de données indépendants correspondant aux

seize mots du texte clair. Pour effectuer des relevés différentiels il faut récupérer une paire de couples texte clair/résultat intermédiaire, où seul le *least significant bit* (LSB) du mot visé change d'un texte clair à l'autre. Si la distance de Hamming trouvée entre les deux résultats intermédiaires d'une paire est 9, 12, 23 ou 24, une seule paire de données en entrée du tour AES peut en être à l'origine. Un simple OU exclusif entre un des textes clairs et la valeur déduite en entrée des Sbox permet de retrouver le mot correspondant de la clé. Si la distance de Hamming est différente de ces quatre valeurs, il faut renouveler l'opération en choisissant une nouvelle paire. En répétant ces opérations pour chacun des mots la clé peut être récupérée. Considérant 128 paires possibles pour chacun des seize mots il faut entre 16 et 2 000 paires pour trouver la clé, avec une espérance statistique de 412,8 paires.

I.3.1. Protection des chaînes de scan

Les chaînes de scan ont rapidement été identifiées comme une faiblesse des circuits du point de vue de la sécurité. Pour avoir des circuits sécurisés, la plus simple des solutions est de se passer de circuit de test, ce qui devient de moins en moins réaliste au fur et à mesure de l'évolution de la complexité des circuits. Une autre solution est de rendre inaccessible la chaine de scan une fois le test de fin de fabrication terminé au moyen de fusibles [19]. Cependant l'utilisation de cette technique s'estompe devant les nouvelles applications utilisant durant la vie du circuit ces infrastructures, notamment celles détaillées en partie I.1.4. Par ailleurs, certains attaquants ont les moyens de détecter un fusible déconnecté, et de le reconnecter, rendant cette protection illusoire. Différentes techniques de sécurisation remplaçant la déconnexion sont présentées dans la suite de cette section.

I.3.2. Protections inhérentes

Les protections dites inhérentes n'ont pas été conçues au départ comme tel, il s'agit de techniques développées pour rendre les infrastructures de test plus efficaces. Des chercheurs ont par la suite étudié l'impact de ces méthodes sur la sécurité [20] [31]. Lorsque le circuit possède plusieurs chaînes de scan, un compacteur de chaîne peut être utilisé pour faciliter l'utilisation. Il peut s'agir par exemple d'un OU exclusif comme représenté sur la Figure I.8. Ces éléments diminuent l'observabilité du circuit et par conséquent rendent les attaques plus difficiles à entreprendre. Cependant, il est toujours possible de réussir à localiser des registres et mener des attaques [24].

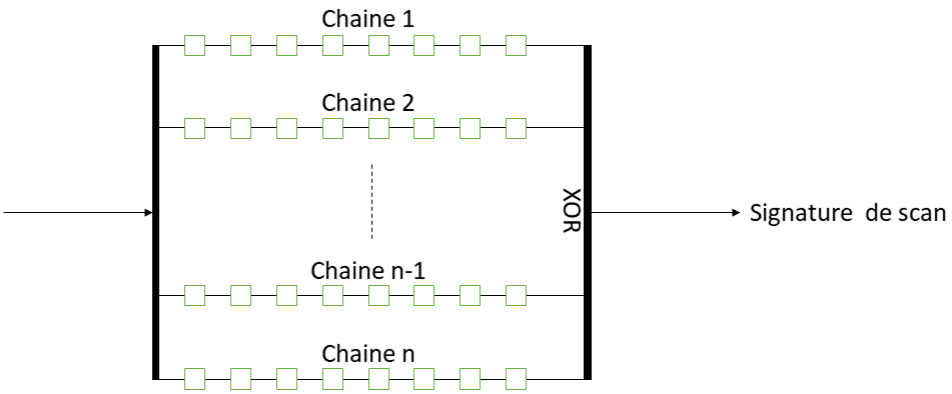


Figure I.8 Chaînes de scan multiples avec compacteur

Une autre solution est l'utilisation de mécanismes d'autotest ou BIST, conçus pour automatiser le test d'une partie du circuit. Il génère lui-même les vecteurs de test, souvent grâce à un générateur pseudo aléatoire implémenté sous forme de *Linear Feedback Shift Register* (LFSR), et compile les réponses en forme de signature. Ensuite, comparé à une signature de référence, le BIST est capable de vérifier si le circuit est en bon état ou non, et de communiquer cette information via la chaîne de scan sans fournir de détails sur les données analysées. Ces informations ne permettent pas de mener une attaque sur l'entité testée par le BIST. Cependant, l'utilisation des BIST supprime la fonction diagnostique proposée par les infrastructures de test plus conventionnelles. Dans les cas où les diagnostiques ne sont pas nécessaires la protection par BIST peut être considérée comme viable [20], même si elle n'est pas applicable sur tous les types de blocs avec une efficacité suffisante.

I.3.3. Protection des données

Plusieurs techniques ont été créées afin de rendre les attaques par chaîne de scan plus difficiles. Par exemple, en découpant en plusieurs segments les chaînes d'un circuit dans les parties devant être protégées, il est possible de créer une méthode de brouillage des vecteurs de test en intervertissant l'ordre de ces segments [33]. La Figure I.9 représente un exemple de cette méthode à trois segments dont l'ordre peut être changé. Lorsque que le mode sécurisé est activé, les segments sont mis dans leur ordre normal, dans l'exemple 1,2 puis 3. Autrement, l'ordre des segments varie à une fréquence déterminée de manière aléatoire ou pseudo aléatoire. La question de l'activation du mode sécurisé est laissée ouverte par les auteurs, qui précisent cependant que ce mode peut être désactivé définitivement grâce à des fusibles. Cette méthode rend les attaques plus difficiles car les comparaisons bit à bit n'ont plus de sens.

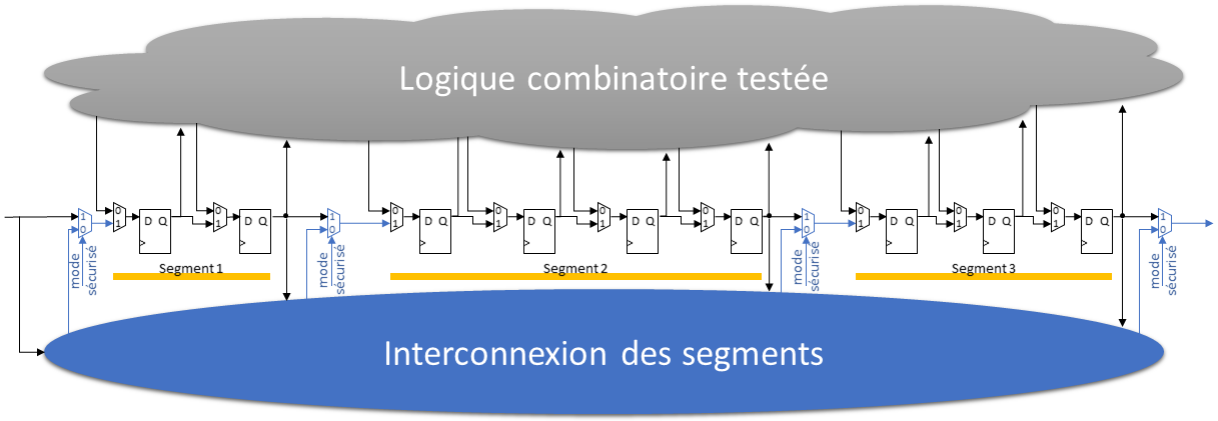


Figure I.9 Chaîne de scan avec brouillage

Pour protéger les clés des chiffreurs il est aussi possible d'effectuer les tests sans que les données sensibles ne soient présentes. Une solution est avancée pour protéger un chiffreur AES dans [34]. A l'initialisation du mode de test, la clé normalement utilisée est remplacée par une séquence de bits issue d'un LFSR. Si l'utilisateur veut utiliser la vraie clé du circuit, il doit insérer en entrée de l'AES la même valeur que celle fournie par le LFSR. Sur le papier cette stratégie permet seulement aux utilisateurs de confiance d'effectuer les tests avec la clé secrète, cependant le protocole pose un problème de sécurité car il prévoit de faire transiter les informations de déverrouillage en clair sur la chaîne de scan. Si les échanges sont espionnés, le système peut ensuite être déverrouillé par une attaque par rejeu. De plus, une attaque différentielle peut être menée sur l'AES pour récupérer, à un certain cycle, la valeur issue du LFSR permettant le déverrouillage du système. Il est ensuite possible d'effectuer la même attaque sur l'AES en visant cette fois la vraie clé secrète. Cette technique ne fera que multiplier par deux le temps d'attaque.

I.3.4. Gestion d'accès

Un autre type de solution est de n'autoriser l'accès aux réseaux de test des zones sensibles qu'aux utilisateurs de confiance. Il a été proposé de créer une extension de sécurité au standard IEEE 1149.1 pour autoriser ou non l'accès aux instructions du contrôleur TAP [35]. L'utilisateur doit insérer une clé secrète dans un registre qui sera ensuite comparé à une valeur stockée en mémoire. Le problème de cette technique est que la clé est transmise en clair sur la chaîne de scan, elle peut donc facilement être interceptée et comme pour la dernière technique de la section I.3.3 être vulnérable aux attaques par rejeu

Une approche étend ce principe au standard IEEE 1687-2014 en protégeant individuellement les segments de test grâce à une modification du SIB, nommée LSIB pour *Locking Segment Insertion Bit* [36] [37]. Il s'agit d'un SIB dont le changement d'état est soumis à une condition supplémentaire. Il y a une unique combinaison de n bits pour chaque LSIB permettant son changement d'état. Cette combinaison correspond à la clé secrète du système dont les registres de contrôle sont répartis et dissimulés dans la chaîne de scan comme représenté sur la Figure

I.10. Les LSIB permettent de sécuriser des sous-réseaux de scan. Les LSIB ont l'avantage de pourvoir fournir différents niveaux d'accès aux utilisateurs du circuit, mais comme pour la précédente technique, l'interception des échanges de connexion donne immédiatement accès aux clés secrètes du circuit. De plus, les clés ne pouvant varier ni d'un circuit à l'autre ni dans le temps, la compromission d'un circuit rend tous les autres vulnérables.

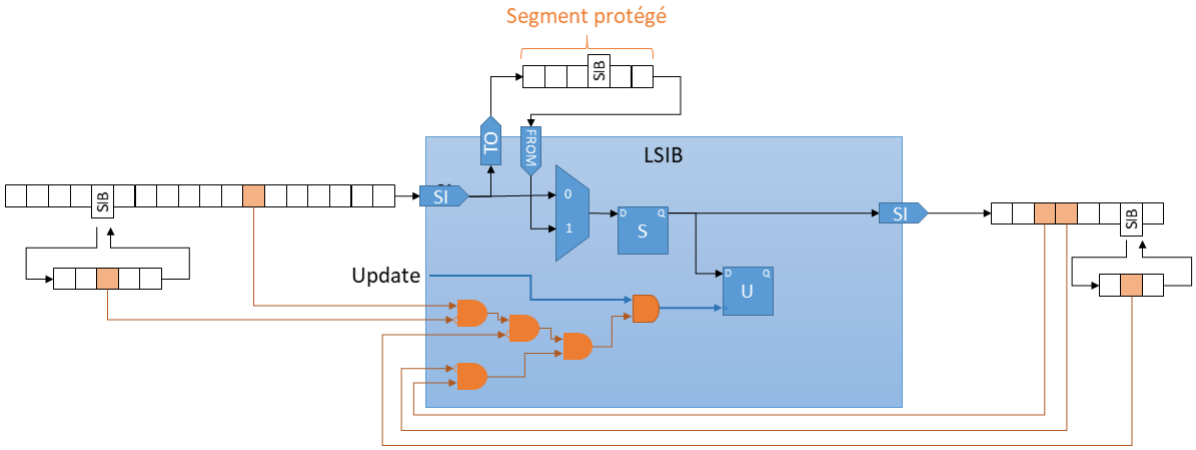


Figure I.10 Schéma contextuel du LSIB

Il est également possible, pour un réseau dont certains segments doivent être protégés, d'en interdire l'accès depuis l'interface TAP [38]. Pour cela un filtre de séquence est placé entre le TAP contrôleur et le RSN, qui analyse les données insérées sur la chaîne. Le filtre de séquence contient une Machine à Etats (MAE) ou *Finite State Machine* (FSM) avec quatre entrées du contrôleur TAP et une seule sortie : *allow*. La MAE observe les opérations de scan et tant que la séquence correspond à un accès restreint, le signal *allow* est actif, sinon la MAE entre en deadlock et le signal *allow* est désactivé, ce qui rend impossible l'accès aux registres protégés. Le signal *allow* conditionne la propagation interne du signal update du RSN : s'il n'est pas actif, il est impossible de changer l'état du réseau. Les auteurs proposent également un algorithme pour générer automatiquement cette MAE. Si cette machine à états devait gérer l'ensemble possible des états et transitions du circuit elle deviendrait beaucoup trop grande, c'est pourquoi ses concepteurs ont choisi de la restreindre à un sous ensemble des états possibles, contraignant ainsi l'utilisation du RSN.

La gestion des accès dans la chaîne de scan interne est un défi qui a été plusieurs fois soulevé, pas seulement par les deux exemples précédents. Celui qui est illustré par la Figure I.11 est similaire au LSIB, basé sur modification du SIB. Le SIB sécurisé (S²IB) possède une seconde chaîne de scan entre SSI et SSO, qui comporte également un registre à décalage (S2) et un registre de mémorisation d'état (U2). L'état de cette chaîne de scan appelée "sécurisée" est une condition supplémentaire à l'ouverture du SIB : il faut que les registres U1 et U2 soient à 1 pour que le S²IB soit ouvert.

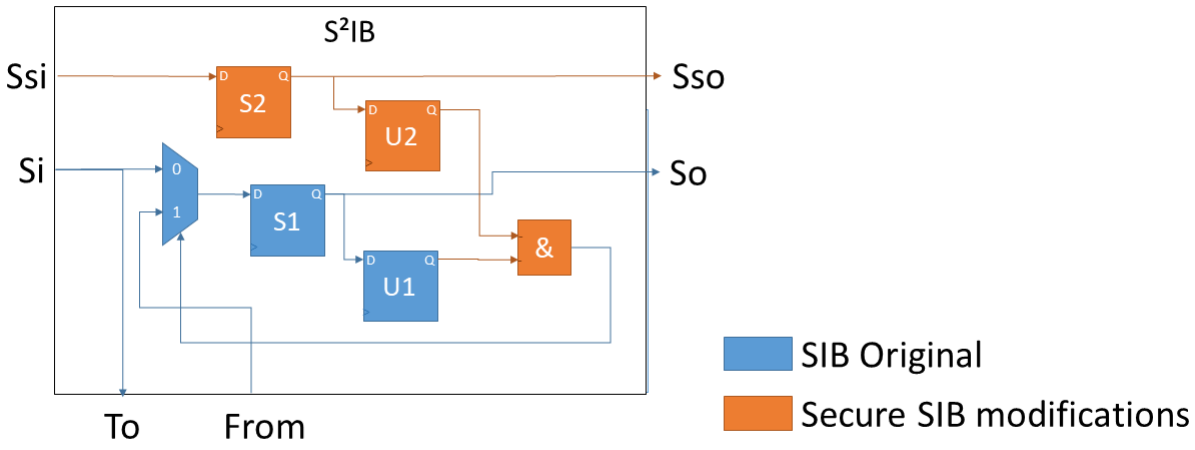


Figure I.11 Schéma du S²IB

Les différents S²IB sont reliés entre eux par une chaîne de scan sécurisée sans hiérarchie. Cette chaîne sert à propager les informations d'autorisation d'accès aux segments aux différents S²IB qu'elle relie. Elle est elle-même pilotée par un contrôleur d'authentification ; la Figure I.12 en est un exemple d'implémentation. En effet en plus de proposer un système de gestion des accès, cette proposition comporte également un volet qui introduit une méthode d'autorisation des utilisateurs.

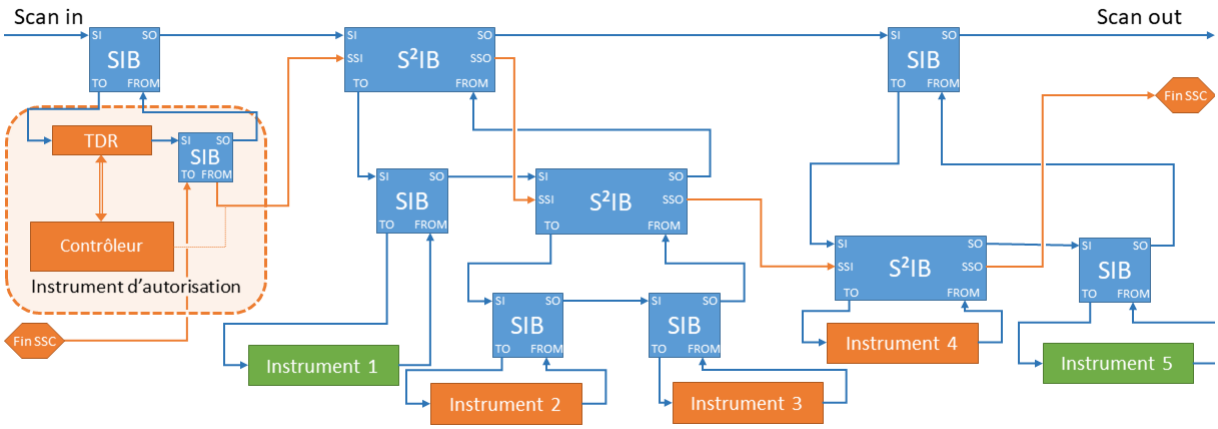


Figure I.12 Contrôle des S²IB

Le contrôleur possède une mémoire qui contient une clé par S²IB. L'utilisateur voulant ouvrir un ou plusieurs S²IB doit connaître la ou les clés correspondantes.

La Figure I.13 montre le protocole d'autorisation :

- Ouverture du SIB du contrôleur : initialisation du contrôleur. Un défi est alors créé aléatoirement.
- Premier cycle d'échange : l'utilisateur émet une requête contenant la configuration voulue appelée Configuration SSC sur le schéma ; elle

contient la liste des S²IB ciblés. Le défi est simultanément récupéré par l'utilisateur.

- Réponse au défi : Le contrôleur et l'utilisateur répondent simultanément au défi. La réponse au défi utilise un hacheur de type SHA2. La première étape est de concaténer le défi et la clé du premier S²IB ciblé puis de hacher l'ensemble. Le résultat doit ensuite être concaténé avec la clé du S²IB suivant puis haché. Cette opération est répétée pour tous les S²IB ciblés. Le résultat final est la réponse au défi.
- Second cycle d'échange : L'utilisateur envoie la réponse qu'il a calculé.
- Comparaison et autorisation : Le contrôleur compare les deux réponses ; si elles sont identiques, l'accès est autorisé grâce à la chaîne de scan sécurisée.

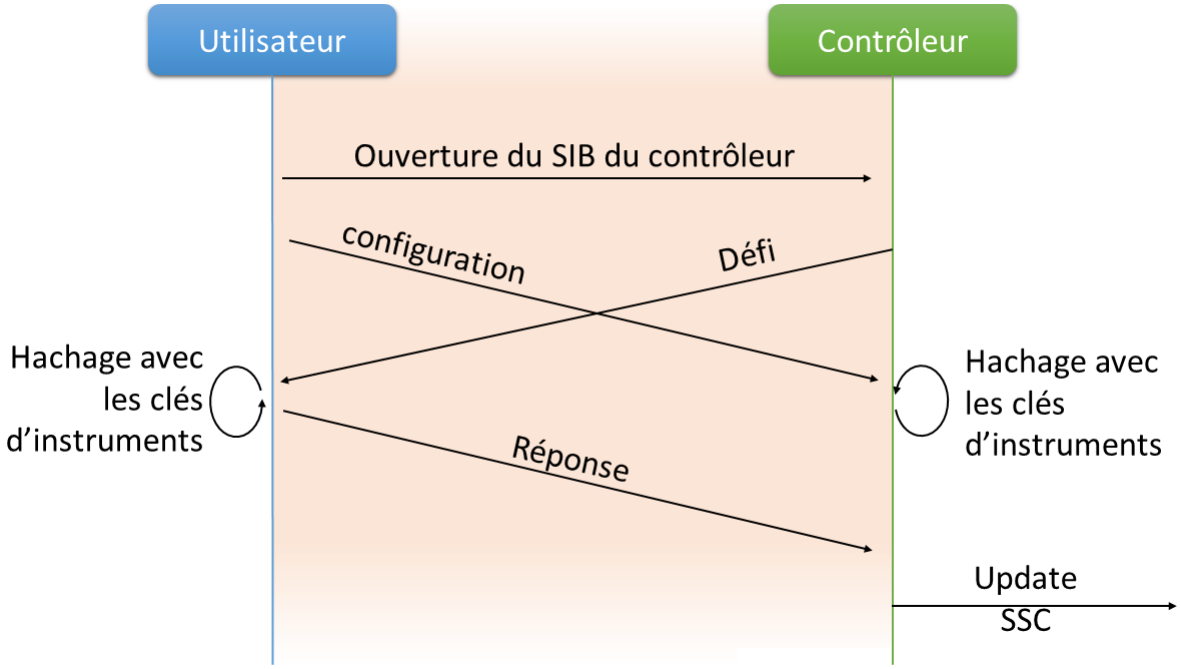


Figure I.13 Protocole d'authentification du contrôleur de S²IB

Cette architecture est très efficace pour gérer l'accès à un grand nombre d'instruments dans un réseau de scan reconfigurable ; elle est même, selon les expérimentations de ses auteurs [39], moins coûteuse en temps de test et en surface d'implémentation, tout en étant plus sécurisé, que les LSIB basés sur les stratégies [36] et [37] pour un ensemble de circuits de test standards [40] lorsque le nombre d'instruments protégés dépasse 8. Le protocole employé n'échangeant jamais les clés, il ne souffre pas de vulnérabilité à l'interception des échanges.

I.3.5. Chiffrement des chaînes de scan

Même si un circuit possède des systèmes de protection nécessaires à l'accès aux segments critiques de la chaîne de scan, les données échangées par la suite

sont toujours susceptibles d'être espionnées ou altérées. Pour garantir l'intégrité et la confidentialité des vecteurs de test, des solutions de chiffrement sont apparues.

Pour protéger leurs FPGA, des constructeurs ont recours au chiffrement des bitstreams [41]. Certaines parties des architectures déployées sur FPGA peuvent contenir des informations secrètes, notamment des éléments propriétaires développés et distribués par les fabricants. Le fait de chiffrer les bitstream et de les déchiffrer à l'entrée du circuit, protège ces propriétés intellectuelles et même empêche leur altération quand le chiffrement est accompagné d'un système de contrôle d'erreur comme un bit de parité ou un Contrôle de Redondance Cyclique (CRC).

En utilisant la même approche, il est possible de protéger les données de test, avec des chiffreurs par bloc [42] ou des chiffreurs de flux [43]. Ces deux propositions répondent au même principe de base, illustré par la Figure I.14 : les vecteurs d'entrée sont chiffrés par l'utilisateur avant d'être insérés dans le canal de communication qui les relie au circuit, les vecteurs voyagent ainsi en sécurité même si le canal n'est pas considéré comme sûr. Ils sont ensuite déchiffrés sur le circuit, par conséquent toutes les données circulent en clair dans la chaîne de scan à l'intérieur du circuit. Avant d'être renvoyées à l'utilisateur, les données sont à nouveau chiffrées ; il ne reste plus à l'utilisateur qu'à déchiffrer les vecteurs pour pouvoir les utiliser.

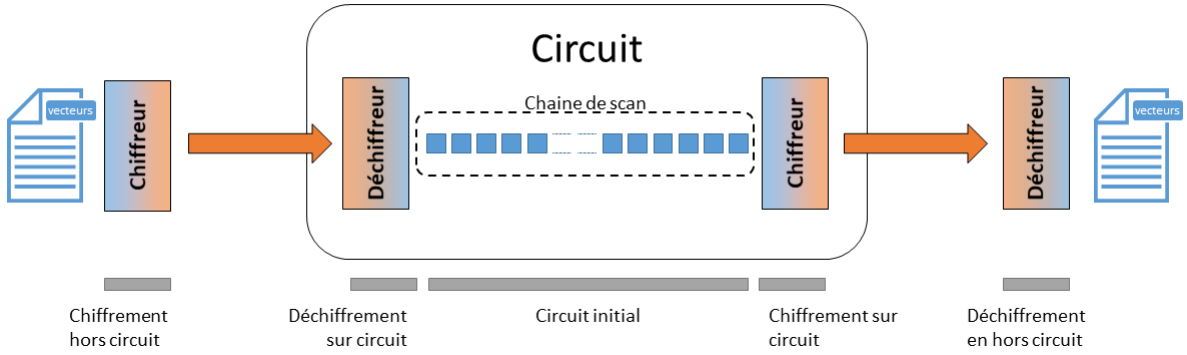


Figure I.14 Schéma de principe du chiffrement de chaîne de scan

I.4. Conclusion

Les infrastructures de test ont su évoluer pour permettre d'assurer le test des circuits intégrés dont la taille et la complexité croissent continuellement. Cette évolution a en outre permis de remplir des fonctions telles que le diagnostic, le débogage, etc. Cependant les nouvelles infrastructures représentent également des opportunités d'attaques supplémentaires venant s'ajouter aux nombreuses méthodes déjà existantes.

Pour pallier aux attaques exploitant les RSN, plusieurs stratégies ont été mises en place. Il est possible d'utiliser des fonctions inhérentes comme le BIST ou la compression de chaîne, rendant les attaques plus difficiles mais réduisant par la

même les fonctionnalités autres que le test. D'autres méthodes permettent de gérer les accès à l'intérieur des RSN et de chiffrer les vecteurs de tests avec chacune leurs forces et limitations. A partir des éléments soulevés jusqu'ici, le chapitre II présente une méthode cherchant à pallier certaines limitations des approches qui ont été présentées.

Chapitre II.
Conception d'un système de sécurisation d'infrastructure de test

La thèse a en partie été financée par le projet européen HADES pour répondre à la problématique des accès sécurisés dans les RSN dans les circuits de l'internet des objets (IoT) qui doivent être petits, peu coûteux et économes en énergie.

Ce chapitre décrit la conception d'un système de sécurité pour le test. Dans une première section, le projet HADES sera brièvement présenté ainsi que ses ambitions en matière de test sécurisé. Une synthèse des différentes techniques répondant au moins partiellement aux objectifs de HADES en matière de test sécurisé est l'objet de la deuxième section. La troisième section est destinée à la gestion et à la distribution des clés secrètes. La structure et la description des différents éléments composant le système font l'objet de la quatrième partie. La cinquième section présente et explique le protocole d'authentification. L'ajout d'une technique de chiffrement au système est présenté dans la sixième et dernière section.

II.1. Projet HADES

Le projet HADES a pour intention de développer une infrastructure de test hiérarchique sécurisée et intelligente pour augmenter les performances et la robustesse des systèmes de test intégrés. La partie du projet qui concerne ce travail de thèse vise à améliorer la sécurité dans les chaînes de scan en prohibant les accès illégitimes aux parties critiques du circuit. Les exigences de cette partie forment les objectifs et les contraintes du système présenté dans ce chapitre.

Le système doit proposer une authentification mutuelle en utilisant un protocole de défi réponse. Il doit rendre l'infrastructure de test capable d'utiliser des vecteurs chiffrés pour garantir leur confidentialité et au moyen d'un CRC de vérifier leur intégrité. Pour éviter que la fuite de la clé secrète de l'un des circuits ne compromette la sécurité de tous, les clés utilisées doivent pouvoir être différenciées d'un circuit à l'autre et être mises à jour. Différents niveaux d'accès doivent être possibles en fonction des autorisations données aux différents utilisateurs.

Pour réaliser ce système de sécurité, ces objectifs ont été répartis entre deux thèses, l'une consacrée à l'intégrité et la confidentialité des données, l'autre, présentée ici, a comme objectif de gérer les accès à l'intérieur des RSN. En plus des objectifs cités, le système doit pouvoir être intégré dans des IoT, c'est pourquoi il doit être optimisé pour avoir un petit budget énergétique et une surface d'implémentation maîtrisée. Certaines approches vues dans la littérature offrent des solutions qui atteignent plusieurs de ces points, mais pas leur totalité.

II.2. Synthèse des méthodes de gestion d'accès

Deux solutions de l'état de l'art se démarquent car elles répondent à la volonté de pouvoir fournir différents niveaux d'accès aux utilisateurs et sont capables de gérer des accès à l'intérieur du RSN. Comme requis, le *Fine Grained Access* (FGA) [39] utilise une méthode d'authentification des utilisateurs pour accorder les accès. Cette méthode basée sur un protocole défi-réponse n'est en revanche pas utilisée par la seconde stratégie, la stratégie LSIB [36]. Aucune des deux solutions ne permet en l'état la modification des clés d'accès. Pour le FGA il serait possible de rendre les clés secrètes modifiables en remplaçant la mémoire ROM par une *Electrically-erasable programmable read-only memory* (EEPROM), cependant ce changement augmenterait considérablement le coût et la surface d'implémentation du système, compte-tenu notamment de la mémorisation d'une clé par instrument protégé.

Au sujet de la performance et du coût dans la chaîne de scan elle-même, il s'agit de comparer ces deux solutions entre elles grâce aux résultats publiés dans [39] ainsi qu'une étude menée pendant cette thèse. La méthodologie de notre étude est davantage détaillée dans la section IV.3 dédiée à l'étude d'impact, où des résultats ont été obtenus en suivant le même protocole. L'objectif ici est de situer les limites de chaque approche. La Figure II.1 illustre les résultats d'une implantation des systèmes de sécurisation par accès LSIB avec des clés de 80 bits et par utilisation de l'approche FGA. L'utilisation de LSIB permet d'avoir un surcoût proportionnel au nombre de protections à apporter, ce qui permet d'être compétitif pour un petit nombre de protections. A l'inverse, le FGA nécessite un coût d'implémentation fixe élevé qui correspond à la partie du contrôleur et du cryptoprocesseur et une plus faible partie dépendant du nombre de protections qui correspond à l'introduction des S²IB et à la mémoire des clés d'instrument.

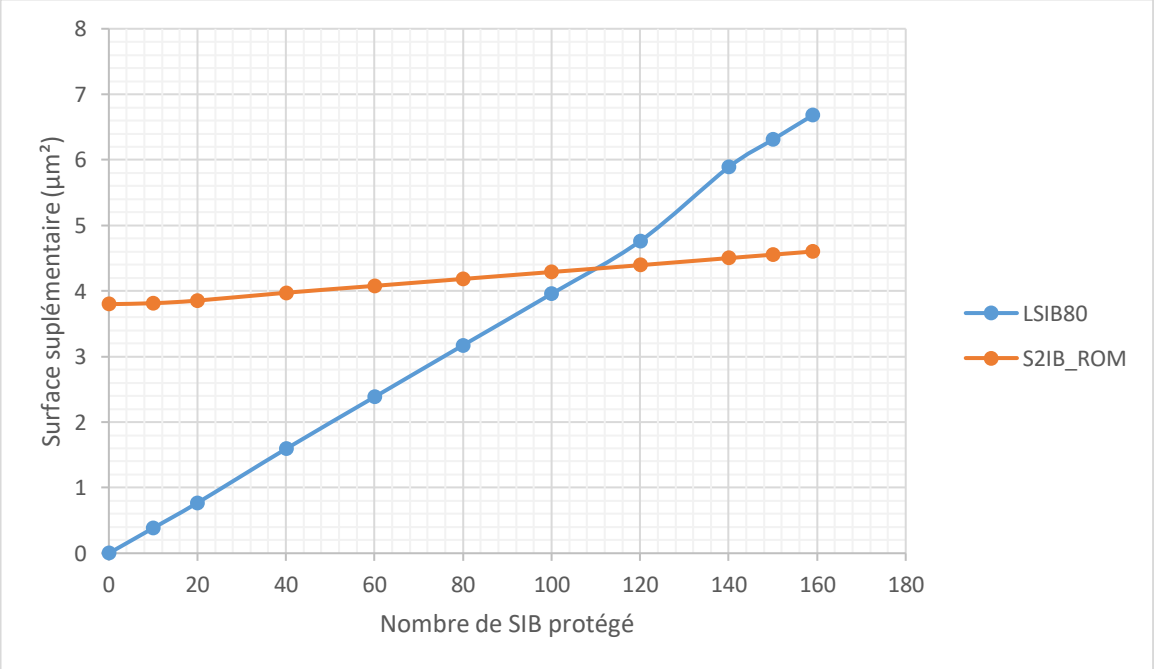


Figure II.1 Surface d'implémentation supplémentaire de LSIB et FGA sur le circuit de test T512505

La Figure II.2 montre le temps nécessaire pour ouvrir tous les SIB du même circuit en fonction du nombre de SIB protégés pour les technologies LSIB et FGA. À l'instar des surfaces d'implémentation, l'utilisation de LSIB a une composante proportionnelle importante. Le temps d'ouverture avec FGA augmente faiblement en fonction du nombre de sites protégés et la composante constante est relativement faible. Dans ce cas précis, la technologie LSIB est plus rapide seulement lorsque le nombre de SIB protégés est très petit, autrement c'est FGA le plus rapide. Il faut tout de même prendre en considération que le temps d'ouverture dépend aussi fortement de la topologie du réseau de test.

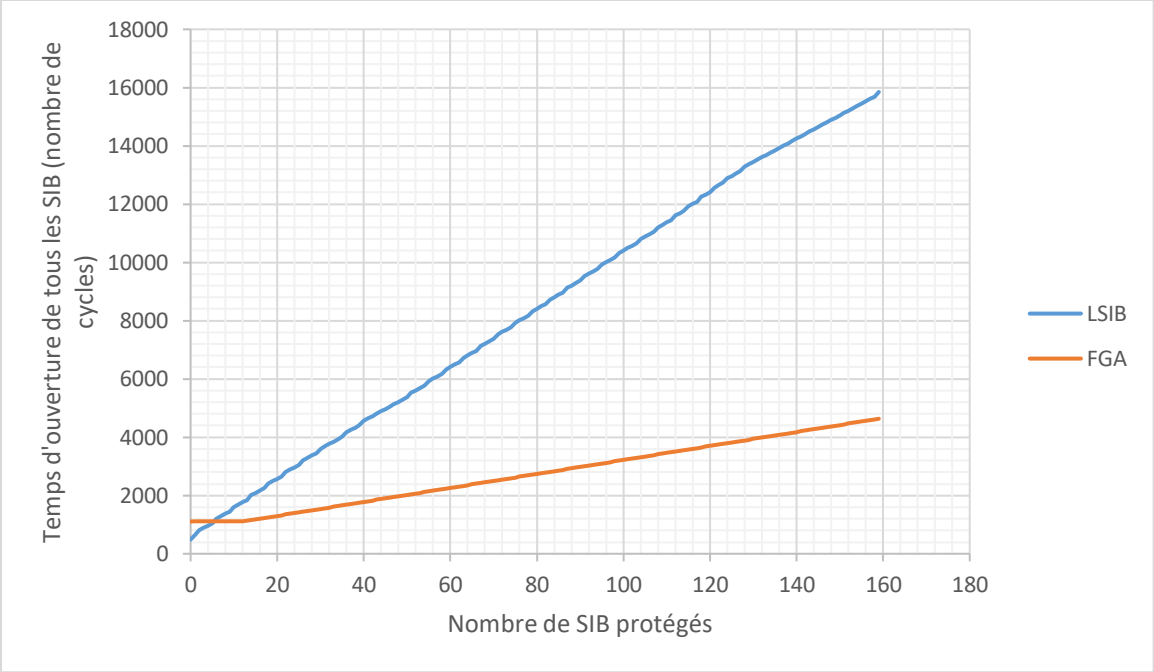
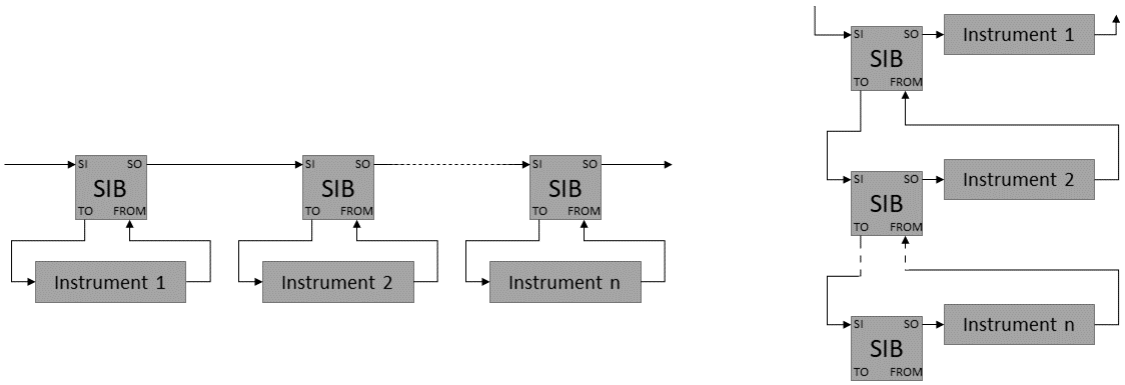


Figure II.2 Temps d'ouverture de tous les SIB en fonction du nombre de SIB protégées

Pour rendre plus visibles les effets des technologies, les résultats de deux topologies aux antipodes sont présentés. Ces deux topologies sont décrites par la Figure II.3. La topologie en ligne permet d'accéder directement à tous les SIB de la chaîne de scan, tandis qu'avec la topologie en colonne pour ouvrir un SIB il est nécessaire d'ouvrir tous ceux qui le précèdent dans la chaîne. Les temps d'ouverture reportés en Figure II.4 montrent que lorsque la chaîne est organisée en ligne, les stratégies LSIB et FGA ne diffèrent que peu. En revanche dans le cas d'une organisation en colonne utiliser des LSIB pour des grands circuits ralentit beaucoup plus le test que l'utilisation de FGA, comme l'illustre la Figure II.5.



Topologie en ligne

Topologie en colone

Figure II.3 Représentation des topologies en ligne et en colonne

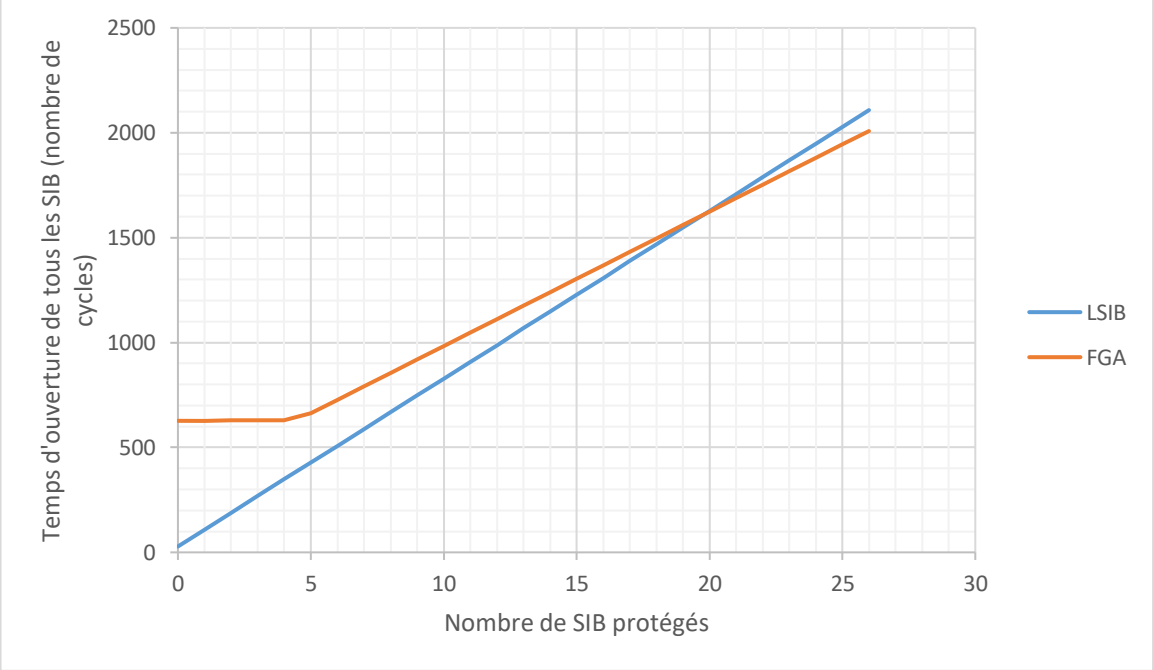


Figure II.4 Temps d'ouverture des SIB avec une topologie en ligne

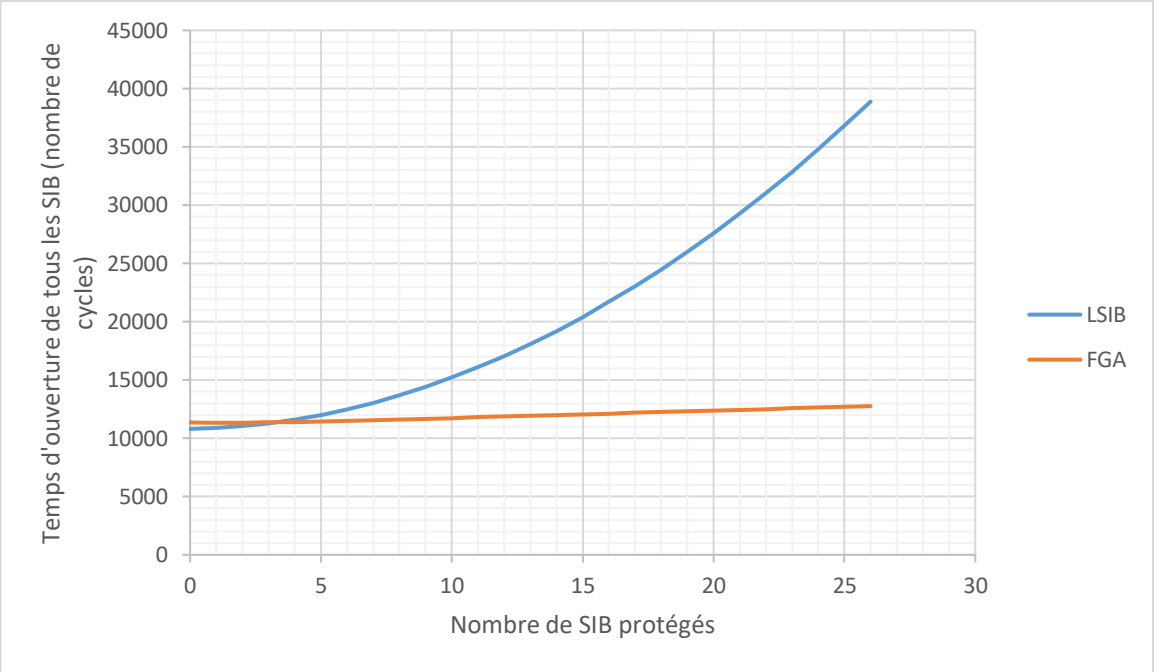


Figure II.5 Temps d'ouverture des SIB avec une topologie en colonne

Les temps d'accès sont équivalents pour des petits circuits, le FGA est le plus rapide pour les grands circuits, bien que le temps d'accès augmente proportionnellement au nombre de S²IB. De même, les coûts d'implémentation varient en fonction du nombre de SIB protégés (LSIB et S²IB). Choisir des LSIB sera moins cher pour des petits circuits et le FGA est plus compétitif pour les grands circuits.

Concernant la gestion des accès dans les infrastructures de test, le Tableau II.1 compare les solutions de [36] et [39] aux objectifs et contraintes avancés par HADES. Les termes de grand circuit et de petit circuit y désignent respectivement les circuits avec beaucoup de segments protégés et ceux qui en ont peu.

Tableau II.1 Avantages et inconvénients des systèmes d'accès existant

	Locking SIB(LSIB)	**Fine Grained Access (FGA)**
Accès authentifié	Non	Oui
Plusieurs niveaux d'accès	Oui	Oui
Clés modifiables	Non	Non
Vitesse d'authentification	Equivalent pour les petits et grands circuits	Plus rapide pour les grands circuits
Surface d'implémentation	Plus avantageuse pour les petits circuit	Plus avantageuse pour les grands circuits

Les comparaisons présentées ici ont permis d'orienter l'élaboration du système de sécurité dont les différents aspects sont présentés au long de ce chapitre. Même s'il est vrai que les performances et les coûts ne donnent pas une préférence évidente pour l'un ou l'autre des systèmes, l'architecture proposée par FGA est la plus proche des objectifs car elle offre notamment un système d'authentification forte de l'utilisateur.

II.3. Politique de gestion de clé

La politique de gestion de clé dans cette proposition est centrale ; il en découle une grande partie des orientations techniques et du protocole d'authentification. C'est pourquoi dès cette partie, les choix concernant les différents objectifs présentés en partie II.1 doivent être exposés.

II.3.1. Configurations et clés de configuration

Dans le contexte de la réduction de consommation des dispositifs électroniques, particulièrement présent dans notre contexte des IoT, il est important de réduire au maximum le nombre de calculs et d'opérations lors de l'exécution d'une tâche car ils représentent une grande partie de la consommation : la consommation dynamique. L'autre partie de la consommation est la consommation statique qui est directement liée à la taille du circuit. En somme, s'il est possible de réduire le temps d'authentification en réduisant le nombre d'opérations d'une part et en réduisant la taille du circuit d'autre part, cela aura également pour effet de diminuer la consommation.

Pour réduire le nombre d'opérations à effectuer pour accorder l'accès à un ensemble de segments, au lieu d'utiliser des clés d'instrument comme FGA [39], chaque configuration possible sera liée à une clé appelée *Segment Set Authorization Key* (SSAK). Ainsi pour accéder n'importe quelle combinaison de segment, une seule clé, et donc une seule opération sera nécessaire pour résoudre le défi. La Figure II.6 montre un exemple de configuration. A chaque S^2IB correspond un bit rangé dans l'ordre de la chaîne de scan sécurisée dans la partie « ensemble segment » (« *segment set* ») du vecteur de configuration. Cet ensemble de segments forme les bits de poids faible du vecteur de configuration. A ce stade les autres bits du vecteur ne sont pas utilisés et donc laissés à zéro.

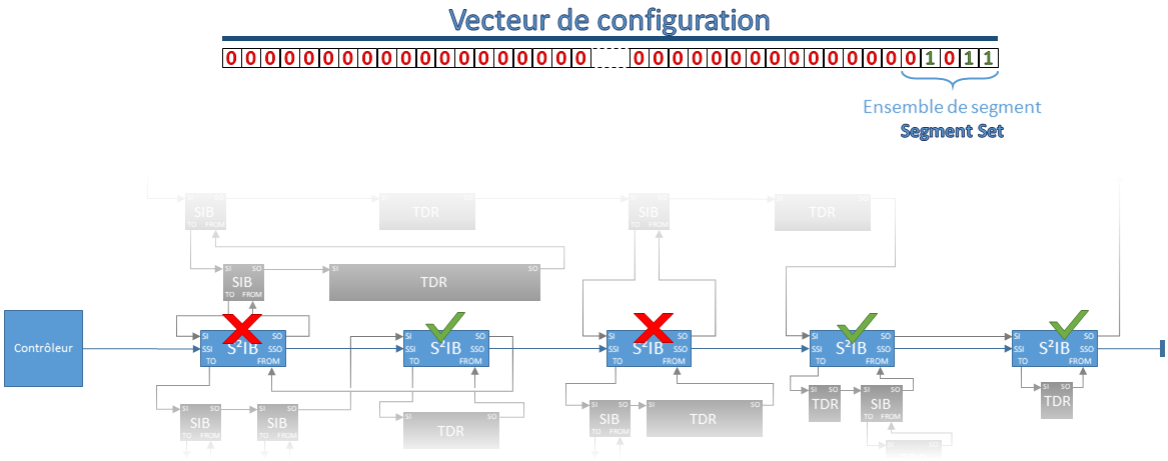


Figure II.6 Vecteur de numérotation et ensemble de segment

II.3.2. Génération procédurale de clé

Le grand nombre de clés de configuration est potentiellement un problème majeur. En effet, s'il ne faut que n clés d'instrument pour gérer les accès sécurisés d'un RSN possédant n S²IB, il faut compter jusqu'à 2^n clés de configurations. Ce grand nombre de clés demanderait une mémoire beaucoup trop grande pour le système de sécurité de l'infrastructure de test d'un IoT. C'est pourquoi un système de génération procédurale des clés est employé. Le but est de permettre au système de sécurité de générer n'importe quelle clé de configuration à partir d'une unique clé de circuit. Pour que la sécurité du système soit maintenue, il faut impérativement qu'il soit impossible pour un utilisateur possédant un couple configuration / SSAK de remonter à la clé unique du circuit. Afin de respecter ce critère une fonction cryptographique est utilisée ; il peut s'agir d'un chiffreur ou d'un hacheur. La Figure II.7 présente cette génération pour les deux méthodes. La méthode utilisant le chiffrage par bloc (a) se sert de la clé de circuit comme clé de l'AES (ou d'un autre chiffrement), du vecteur de configuration comme texte clair et le texte chiffré donne la SSAK. Ainsi, il impossible pour un utilisateur connaissant le vecteur de configuration de deviner la clé. De même avec le hacheur SHA256 (b), étant donné les propriétés des hacheurs, il n'est pas possible de remonter à la clé de circuit en connaissant la SSAK et le vecteur de configuration. À l'instar du chiffrage par bloc, le chiffreur par flux (c) utilise la clé de circuit comme clé du chiffreur et le vecteur de configuration comme vecteur d'initialisation du chiffreur. Cependant, il est nécessaire d'ajouter un accumulateur chargé de concaténer les bits de la clé de flux pour former la clé SSAK. Tout comme les deux autres solutions, les propriétés de ce cryptoprocesseur ne permettent pas de remonter à la clé de circuit dans le cas d'une implémentation correcte [45]. Outre les différents types de cryptoprocesseurs, la Figure II.7 montre que la taille des clés et du vecteur de configuration peut être choisie en fonction du niveau de sécurité désiré. Cependant les utilisateurs autorisés sont en possession d'un couple vecteur de configuration / SSAK. Si l'un d'entre eux est mal intentionné il pourra monter une attaque par force brute hors ligne ; il est donc nécessaire de ne pas choisir une taille de clé trop petite.

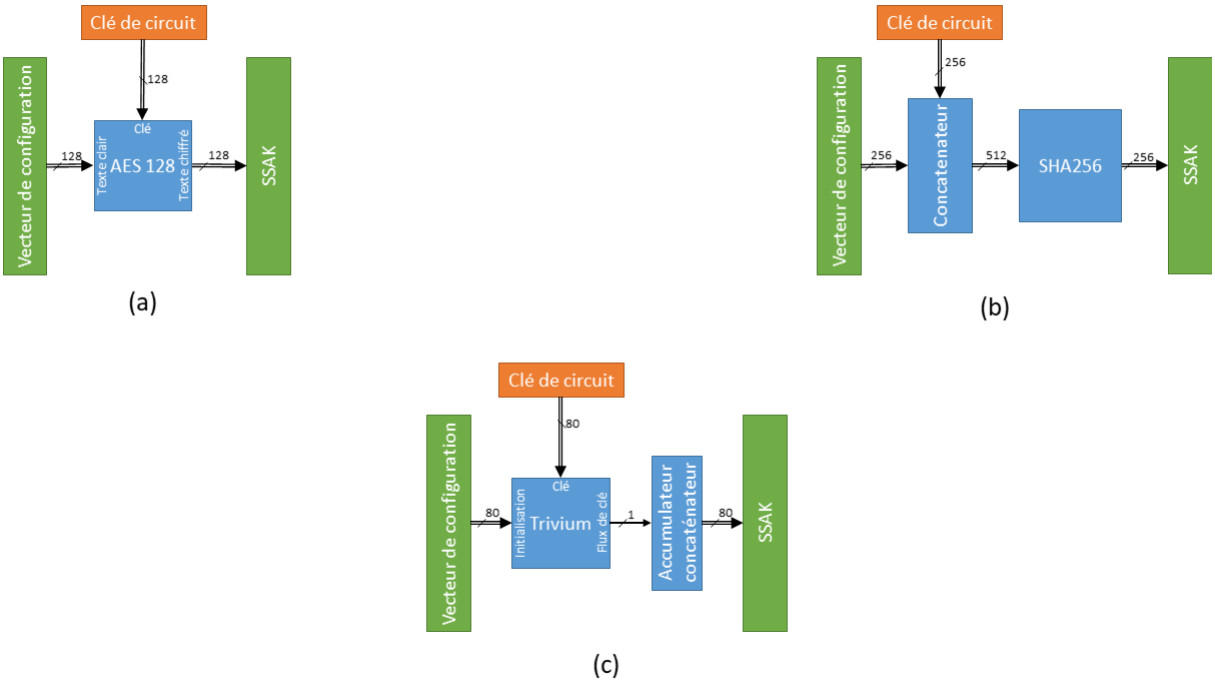


Figure II.7 Génération procédurale de SSAK (a) Avec AES (b) avec SHA256

II.3.3. Clés reconfigurables

Grâce au système de génération procédurale de clé, il n'y a besoin de garder en mémoire qu'une seule clé, ce qui rend le problème de la taille et du coût des mémoires reprogrammables mentionné en II.1 caduc. En effet, il est possible d'utiliser une EEPROM ou toute autre mémoire non volatile réinscriptible contenant une seule clé pour rendre l'ensemble des clés de configuration modifiables.

II.3.4. Distribution

L'un des problèmes récurrents des systèmes de sécurité est la distribution des clés. Dans le cas présent, cela sera assuré par un fournisseur de sécurité comme l'illustre la Figure II.8. Ce tiers de sécurité est chargé de générer la clé de chaque circuit, ensuite programmée dans le registre de clé du contrôleur SSAK. Le fournisseur est également chargé de fournir les identifiants aux utilisateurs. Dans l'exemple de la Figure II.8 il génère le vecteur de configuration d'un utilisateur lambda, en incluant les autorisations que cet utilisateur est sensé recevoir. Il utilise ensuite le même générateur de clé procédurale que celui intégré avec le contrôleur, pour générer la SSAK à partir de la clé de circuit et du vecteur de configuration. Si les circuits possèdent des clés différentes, il faut distribuer des identifiant relatifs à chacun des circuits.

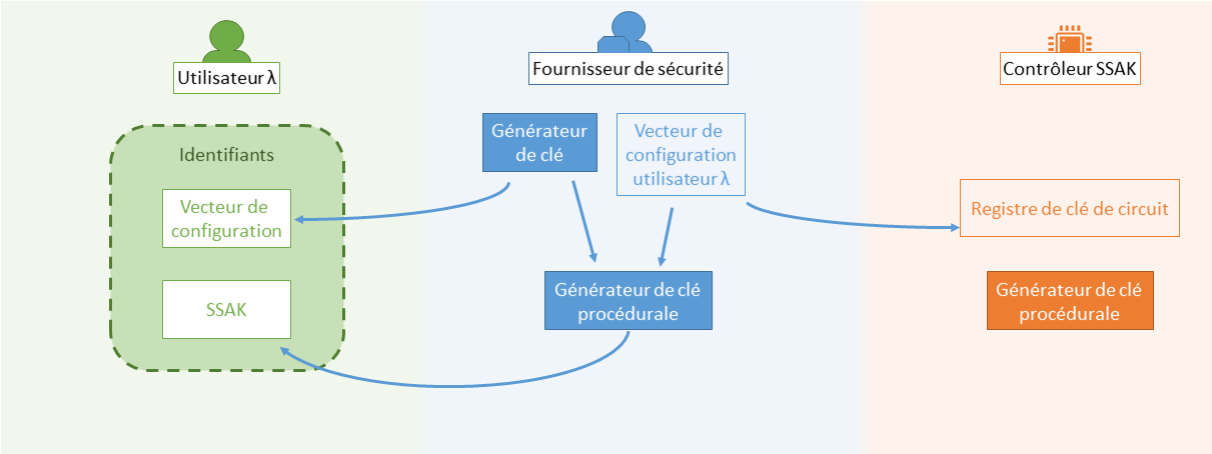


Figure II.8 Distribution des clés et des identifiants

Au-delà de la distribution, d'autres problèmes concernant la gestion des clés et leur stockage existent. En effet il faut être capable de stocker ces informations de manière sécurisée pour éviter les fuites de données et de manière non volatile car ces données ne sont pas censées s'effacer lors d'une remise à zéro du circuit ou lors d'une interruption d'alimentation. Ce point particulier de stockage des clés n'a pas fait l'objet d'une innovation particulière dans le cadre de cette thèse car il est possible de s'appuyer sur les unités de gestion de clé (KMU) présentes sur les circuits possédant des primitives cryptographiques [44], dont certaines peuvent gérer la mise à jour et la révocation de clé.

II.3.5. Identification d'utilisateur

Tel que présenté, le système ne permet pas d'identifier les utilisateurs, mais seulement de vérifier que leurs autorisations soient conformes à l'accès qu'ils demandent. Pour effectuer la vérification d'identité, il est possible comme l'illustre la Figure II.9 d'ajouter un numéro d'identification de l'utilisateur qui rend les clés propres à chaque utilisateur. Cette identification a des applications en termes de suivi de la sécurité. Si un utilisateur malhonnête diffuse ou vend ses identifiants d'autorisation, il sera immédiatement identifiable car son identité figure dans ses identifiants. Une liste noire peut également être utilisée pour que le contrôleur refuse les numéros des utilisateurs dont les identifiants ont fuité. À l'image de la clé de circuit, la gestion de la liste noire peut également être déléguée à une KMU.

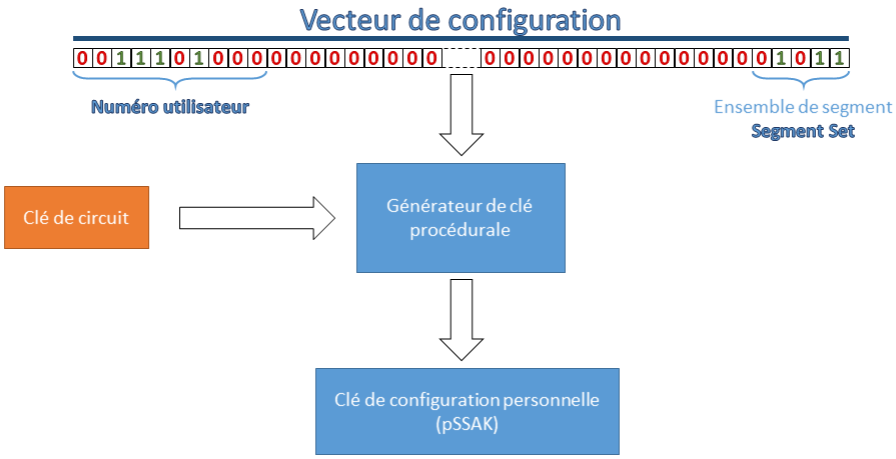


Figure II.9 Clé de configuration personnelle

II.4. Architecture du système

Cette section présente le cœur de l'architecture de la solution. Comme expliqué en II.2, la solution FGA présentée dans [39] est la plus proche des objectifs, c'est pourquoi une partie de son architecture est reprise et adaptée pour le système présenté. Comme le FGA, il fait reposer sa stratégie de déverrouillage des segments grâce à une chaîne de scan sécurisée et des S²IB. En revanche, comme le montre la politique de gestion de clé présentée en II.3, il y a déjà plusieurs modifications de comportement du système auquel le contrôleur SSAK doit s'adapter. Plus largement, cette partie va détailler l'architecture du système SSAK, le contrôleur et ses différents périphériques.

II.4.1. Contrôleur et périphériques

Le système proposé possède plusieurs configurations possibles Il se présente sous la forme d'un contrôleur et de plusieurs périphériques pouvant facilement être remplacés. La Figure II.10 liste les périphériques et fait apparaître les connexions avec le contrôleur. Cette architecture modulaire a été choisie pour permettre de facilement changer de cryptoprocesseur et d'ajouter ou non une liste noire d'utilisateur. L'utilisation d'une connectique standardisée avec les périphériques, ici l'utilisation des bus AXI4--Stream, permet de rendre l'intégration d'un nouveau périphérique plus facile. Pour installer un nouveau cryptoprocesseur, il suffit de lui ajouter une enveloppe AXI4-Stream et le connecter au contrôleur sans avoir à se soucier des problèmes de synchronisation des données. Selon la cible d'implantation choisie, ce bus peut être remplacé par un autre bus standardisé.

Chacun de ces périphériques joue un rôle particulier :

- Le générateur de nombres aléatoires (TRNG)

Le TRNG génère des nombres aléatoires qui sont utilisés pour proposer un défi à l'utilisateur. Généralement les circuits sécurisés possèdent déjà un générateur de nombres aléatoires, il n'est donc pas nécessaire si c'est le cas, de l'implémenter une seconde fois spécifiquement pour cette application.

- Le cryptoprocesseur

Lors de l'authentification le cryptoprocesseur joue deux rôles : Il est chargé de générer la clé SSAK et également de résoudre le défi. Il peut s'agir d'un chiffreur par bloc, d'un hacheur ou d'un chiffreur de flux. Comme pour le TRNG si un cryptoprocesseur est déjà présent dans le circuit il peut être réutilisé.

- Le registre de clé

Ce registre est la mémoire qui contient la clé de circuit. Elle est représentée et implémentée ainsi pour le développement du système mais il s'agit en réalité d'un module de gestion de clé.

- La liste noire

Cette mémoire optionnelle contient la liste des utilisateurs dont l'accès est retiré. Il s'agit d'une méthode permettant de révoquer les droits d'un ou de plusieurs utilisateurs sans avoir à changer la clé de circuit ce qui impliquerait de devoir redistribuer tous les certificats aux utilisateurs non révoqués. La liste noire est implémentée de la même manière que le registre de clés mais sa mission devra également être confiée à une unité de gestion de clé.

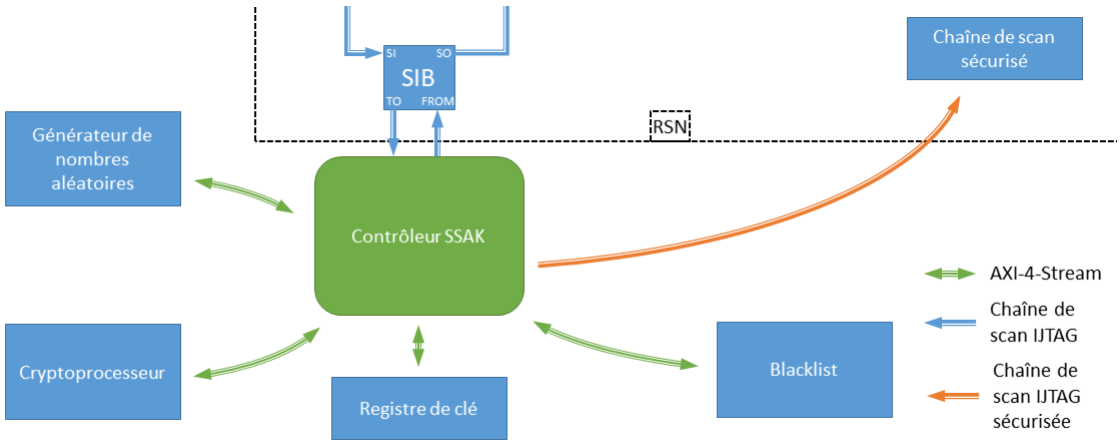


Figure II.10 Contrôleur et ses périphériques

Le contrôle de la Chaîne de Scan Sécurisée (SSC) diffère par rapport à celui du FGA [39] présenté dans la partie I.3.4. La SSC du FGA fait partie du RSN, seul le signal *update* est piloté par le contrôleur qui vérifie le contenu de la chaîne de scan. L'architecture du SSAK à l'inverse propose une SSC totalement indépendante et intégralement pilotée par le contrôleur SSAK, comme l'illustre la Figure II.11. Ce choix est motivé par plusieurs raisons : premièrement, retirer la chaîne de scan sécurisée du RSN rend le protocole plus simple pour l'utilisateur car il ne lui est plus nécessaire d'y gérer l'écriture des données. Deuxièmement, interdire toute action de l'utilisateur sur la SSC peut réduire le risque d'attaques. Enfin, en rendant cette

chaîne indépendante, il est possible de s'en servir pour d'autres usages que l'autorisation ; ce point particulier est décrit dans la section III.2.

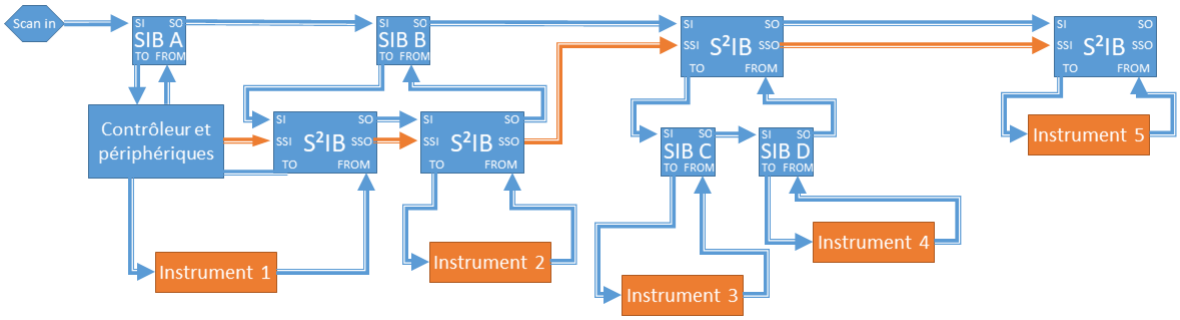


Figure II.11 Contrôleur SSAK et RSN

II.4.2. Contrôleur d'authentification

Les parties précédentes de cette section ont décrit le fonctionnement du contrôleur d'authentification dans son environnement : ses interactions avec ses périphériques, y compris la SSC. Le but de cette partie est d'expliquer le fonctionnement interne du contrôleur. La Figure II.12 est une représentation graphique de l'implantation matérielle. Plusieurs blocs dont une machine à état (FSM, ou MAE) y sont visibles et majoritairement reliés par des bus AXI4-Stream qui permettent d'ajouter des éléments de synchronisation entre les blocs et ainsi d'alléger la MAE. Le contrôleur possède quatre registres dédiés à la configuration, au défi, à la réponse de l'utilisateur et à la SSAK. La réponse et la configuration sont tous deux fournis par l'interface avec la chaîne de scan, le défi est fourni par le générateur de nombres aléatoires, et la SSAK est issue du cryptoprocesseur. Sachant que le cryptoprocesseur doit être utilisé deux fois – une fois pour la génération de la SSAK et l'autre pour la résolution du défi – des multiplexeurs sont placés à son entrée pour que la MAE puisse choisir quelle opération doit être réalisée. Un composant vérifie l'égalité des deux réponses.

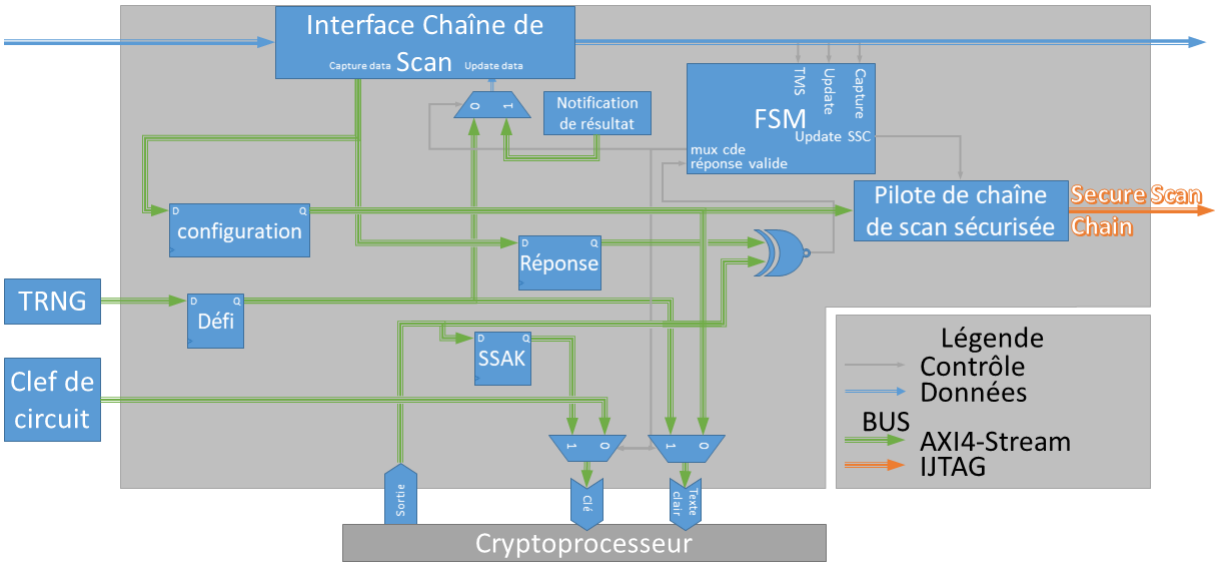


Figure II.12 Implémentation RTL du contrôleur

II.5. Protocole et machine à états

Cette section porte sur le protocole d'authentification. La solution de protection présentée dans ce chapitre utilisant des stratégies issues de la proposition FGA [39], le protocole présenté ici en est issu et est adapté pour être compatible avec les nouvelles approches, notamment la nouvelle politique de gestion des clés. La première partie présente le protocole d'authentification et la seconde présente son implémentation dans la MAE du contrôleur.

II.5.1. Protocole d'authentification

La Figure II.13 représente le protocole décrit ci-après.

L'initiative de l'authentification revient à l'utilisateur. Le protocole démarre lorsque le SIB du contrôleur, nommé SIB A sur la Figure II.11, est ouvert. Une fois le protocole lancé, le contrôleur doit générer un défi grâce au générateur de nombres aléatoires présenté en partie II.4.1. Lors du premier échange de donnée entre l'utilisateur et le contrôleur, l'utilisateur insère dans la chaîne de scan son vecteur de configuration issu de ses identifiants et récupère le défi proposé par le contrôleur.

Ensuite, les deux parties doivent résoudre le défi. Avant de pouvoir répondre au défi, le contrôleur doit générer la SSAK correspondant au vecteur de configuration. La méthode utilisée pour générer cette clé est expliquée dans la partie II.3.2. Une fois que la SSAK est obtenue, il est possible de résoudre le défi grâce à une deuxième utilisation du cryptoprocesseur. Si le cryptoprocesseur est un chiffreur, la SSAK est utilisée comme clé pour chiffrer le défi. S'il s'agit d'un hacheur, la SSAK et le défi seront concaténés puis hachés. Le résultat est conservé par le contrôleur. L'utilisateur étant déjà en possession de la SSAK nécessaire a un travail moindre à fournir. Une opération identique avec la SSAK, le défi et la même primitive

cryptographique doit être effectuée pour calculer sa propre réponse qui est ensuite envoyée au contrôleur.

Facultativement, l'utilisateur peut interroger le contrôleur sur la réussite de l'authentification toujours au travers de la chaîne de scan. La réponse renvoyée par le contrôleur sera concentrée sur les deux bits de poids les plus faibles du segment d'interface. Trois états différents peuvent alors être obtenus : « 00 » signifie que le contrôleur est toujours en train de calculer sa réponse, « 10 » signifie que les deux réponses sont différentes et que l'accès est donc refusé, « 11 » signifie que les deux réponses sont identiques et que l'accès est autorisé. Pour clore le protocole, le SIB du contrôleur doit être fermé.

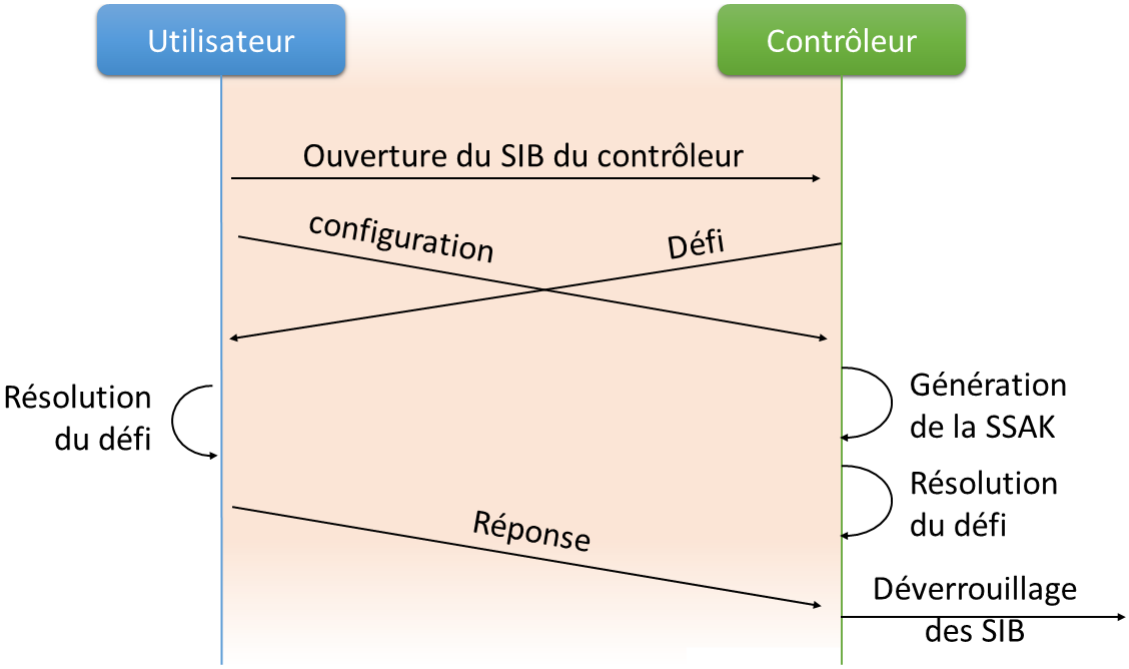


Figure II.13 Protocole d'authentification

Il est à noter que si la partie « ensemble de segment » du vecteur de configuration ne comporte que des « 0 », c'est-à-dire qu'aucun accès n'est demandé, ou que l'annulation de tous les accès en cours est demandée, la demande sera automatiquement acceptée, les étapes suivantes du protocole seront sautées et l'utilisateur pourra immédiatement fermer le SIB du contrôleur.

II.5.2. Machine à états du protocole

Les différents blocs et fonctions à l'intérieur de ce contrôleur sont commandés par une machine à états (présente sur la Figure II.12) et dont le fonctionnement est représenté sur la Figure II.14. A l'état initial, le contrôleur ainsi que la chaîne de scan sécurisée (SSC) sont remis à zéro (RAZ). Lorsque cette étape est active toutes les autorisations d'accès en cours sont révoquées. Tant que les signaux IJTAG capture et select ne sont pas tous deux à « 1 » l'étape RST reste active.

Une fois passés à « 1 », la MAE passe à l'étape « S_CONF » lors de laquelle le vecteur de configuration est enregistré dans son registre dédié. Si aucun accès

n’est demandé dans cette configuration, le contrôleur retournera directement à l’étape « RST », sinon l’étape suivante est atteinte.

C’est lors de l’étape « W_SSAK » que la SSAK est générée par le cryptoprocesseur. Grâce à l’utilisation des bus AXI4-Stream, il suffit à la FSM de maintenir le signal des multiplexeurs du cryptoprocesseur à « 0 » pour que la génération ait lieu. La fin de la génération est attendue avant de passer à l’étape suivante. Durant cette étape les autorisations d’accès sont insérées dans la chaîne de scan.

L’étape « W_RSP » consiste à la génération de la réponse du défi. Comme pour l’étape précédente, il suffit de positionner les multiplexeurs dans l’autre position pour que la génération de réponse se lance ; l’étape suivante est atteinte lorsque la réponse du contrôleur et celle de l’utilisateur sont obtenues.

L’étape « CMP » n’est active que pendant un cycle d’horloge, si les deux réponses sont différentes la MAE repasse en mode « RST » sinon c’est l’étape suivante qui est activée.

L’étape « APP_SSC » consiste en l’application du signal update à la SSC, pendant un seul cycle. Les deux étapes suivantes servent à attendre qu’une nouvelle demande d’authentification soit effectuée pour recommencer le processus.

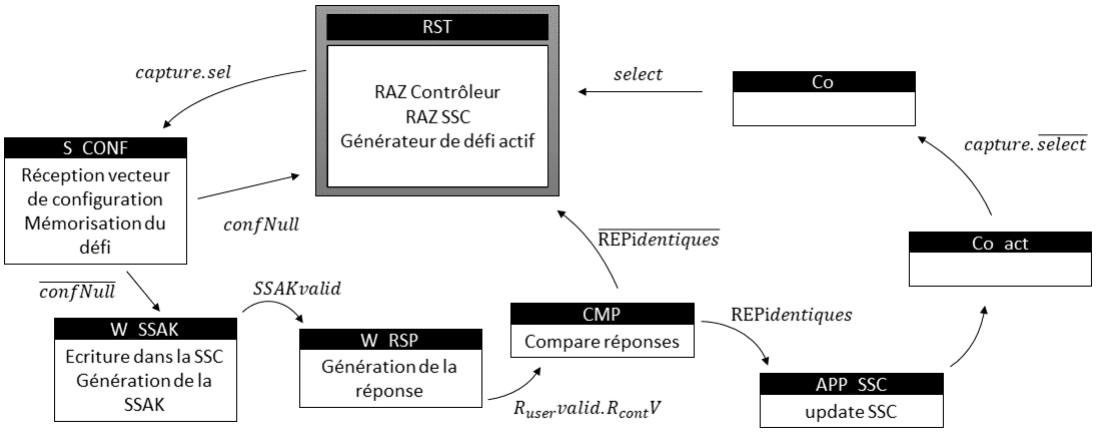


Figure II.14 Machine à états du contrôleur SSAK

Le système présenté dans cette section est développé dans le but de répondre à trois objectifs fonctionnels : il doit autoriser l'accès aux parties critiques du circuit aux utilisateurs autorisés seulement après authentification ; il doit être possible d'avoir un accès personnalisé pour chaque utilisateur ; et il faut être capable de supporter un système de mise à jour de la clé. Si parmi les solutions vues dans l'état de l'art il est possible grâce à la méthode *Fine-Grained Access [39]* d'avoir une authentification par protocole défi-réponse avec des accès personnalisés pour les utilisateurs, cette solution ne propose en revanche pas de moyen optimal pour avoir des clés modifiables. Parce qu’une seule clé secrète est nécessaire dans la proposition SSAK, il est possible d’utiliser une mémoire réinscriptible sans que cela ne soit trop coûteux. Sur ce dernier point, la solution présentée ici prouve qu’elle est

le plus à même à répondre aux objectifs présentés. Cependant pour s'assurer de la pertinence de cette solution, il faut montrer d'une part qu'elle est réalisable, et d'autre part qu'elle est compétitive en termes de coût d'implémentation et d'augmentation du temps de test. C'est dans le Chapitre IV que ces preuves et comparaisons sont présentées.

II.6. Chiffrement des vecteurs

Comme précisé dans la section II.1, la sécurité de l'infrastructure de test que veut proposer HADES est d'une part la gestion des accès dans le RSN et d'autre part la confidentialité et l'intégrité des données. Si la première partie est satisfaite par la conception du contrôleur SSAK, la seconde dépend des travaux d'un autre doctorant. La solution développée pour assurer la confidentialité des données [43] consiste à utiliser un chiffreur de flux pour que l'infrastructure de test puisse utiliser des vecteurs de test chiffrés.

Sans efforts particuliers de fusion de ces deux systèmes, l'infrastructure de test sécurisée d'HADES devrait utiliser deux cryptoprocesseurs, celui utilisé par le SSAK n'étant utilisé que ponctuellement lors de l'authentification d'un utilisateur. En considérant que les cryptoprocesseurs sont la partie la plus coûteuse en termes d'utilisation de surface d'implémentation ainsi que de consommation d'énergie de chacune des deux solutions, il ne serait pas très compétitif de proposer une solution commune comportant deux cryptoprocesseurs différents. C'est pourquoi une solution proposant ces deux fonctionnalités doit être réalisée en mutualisant un maximum de matériel et de protocole.

Le chiffrement des vecteurs proposé dans [43] offre une version modifiée d'un contrôleur TAP pour protéger plusieurs registres notamment la chaîne de scan interne. Il est basé sur le chiffreur de flux TRIVIUM qui nécessite en entrée un vecteur d'initialisation et une clé secrète pour générer des flux de clés. Il est utilisé pour déchiffrer les vecteurs en entrée de la chaîne de scan, et pour chiffrer les réponses à sa sortie. La clé secrète est connue du contrôleur et des utilisateurs tandis que le vecteur d'initialisation est choisi à chaque initialisation (RAZ) de la carte par un générateur de nombres aléatoires. Un registre est ajouté au contrôleur TAP pour pouvoir lire la valeur du vecteur d'initialisation. Il est très important que le vecteur d'initialisation soit différent à chaque utilisation car autrement il est possible pour un attaquant de récupérer la clé de flux avec une attaque "two times pad" [45].

II.6.1. Protocole d'autorisation et de chiffrement

Le protocole de chiffrement des vecteurs prévoit d'être actif en permanence quand le mode de test est activé, même si aucune partie sensible du circuit n'est accédée à travers l'infrastructure de test, ce qui provoque des consommations d'énergie pouvant être évitées. De son côté, le contrôleur d'accès SSAK peut savoir quand un utilisateur cherche à accéder à des parties de la chaîne de scan reconfigurable devant être protégées. Ainsi, en créant un protocole commun, il est

possible de n'utiliser des vecteurs de test chiffrés que si une authentification est effectuée par le contrôleur SSAK.

Le début du protocole commun est identique à celui du SSAK. Pour la suite il faut que les deux parties se mettent d'accord sur une clé de chiffrement. A ce stade, le contrôleur et l'utilisateur possèdent quatre informations en commun : le vecteur de configuration, la SSAK, le défi et la réponse. La SSAK étant la seule n'ayant pas transité dans la chaîne de scan en clair, elle est la seule à pouvoir être considérée comme une clé secrète. Un vecteur de chiffrement de valeur aléatoire doit aussi être choisi. Le défi est certes une valeur aléatoire, mais si elle est choisie comme valeur d'initialisation du générateur de flux de clé, les premiers bits de ce flux correspondront à la réponse du protocole SSAK, déjà échangée en clair sur la chaîne de scan. Le seul choix possible restant est d'utiliser cette réponse comme vecteur d'initialisation. Le chiffrement de flux démarre lorsque s'achève avec succès l'authentification SSAK, et s'arrête sur un RAZ ou un nouvel accès au contrôleur SSAK.

II.6.2. Architecture du système

Pour mettre en œuvre la fusion de ces deux systèmes il est nécessaire de procéder à des changements. Avec cette solution le chiffrement de vecteur n'est plus géré au niveau du contrôleur TAP mais est piloté par le contrôleur SSAK ; la Figure II.15 illustre cette architecture. Une unité de chiffrement contenant un seul cryptoprocesseur est chargée de répondre à la fois aux exigences du protocole SSAK et à la génération du flux de clés. Comme pour l'architecture sans chiffrement, le contrôleur SSAK pilote le cryptoprocesseur, mais ici il a en plus la charge de configurer l'unité de chiffrement une fois l'authentification effectuée pour que le flux puisse être généré.

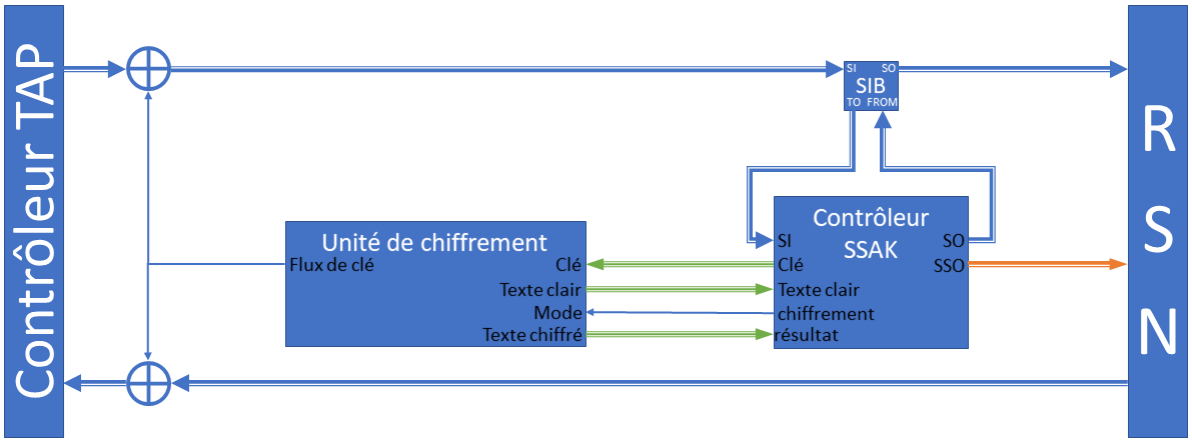


Figure II.15 Chiffrement de chaîne de scan et autorisation SSAK

L'unité de chiffrement est une architecture centrée sur un cryptoprocesseur qui permet de partager sa fonction de chiffrement aux deux applications. La Figure II.16 représente une unité de chiffrement adaptée à un chiffreur par bloc, mais un

hacheur peut tout aussi bien être utilisé. L'unité possède deux modes de fonctionnement : le premier équivaut à un chiffreur par bloc, le deuxième à un générateur de flux de clés. Pour obtenir ce dernier, la sortie du chiffreur est rebouclée sur son entrée pour générer un flot continu de textes chiffrés. Un composant est ensuite chargé de synchroniser le flux de clés issu de ses textes chiffrés avec la chaîne de scan.

Il est également possible d'utiliser un chiffreur de flux pour effectuer les deux opérations. Dans ce cas l'unité de chiffrement s'en trouve changée, la sortie chiffrée étant obtenue en accumulant les bits du flux de clés jusqu'à en obtenir le nombre nécessaire, la génération du flux de clés étant directement effectuée par le chiffreur.

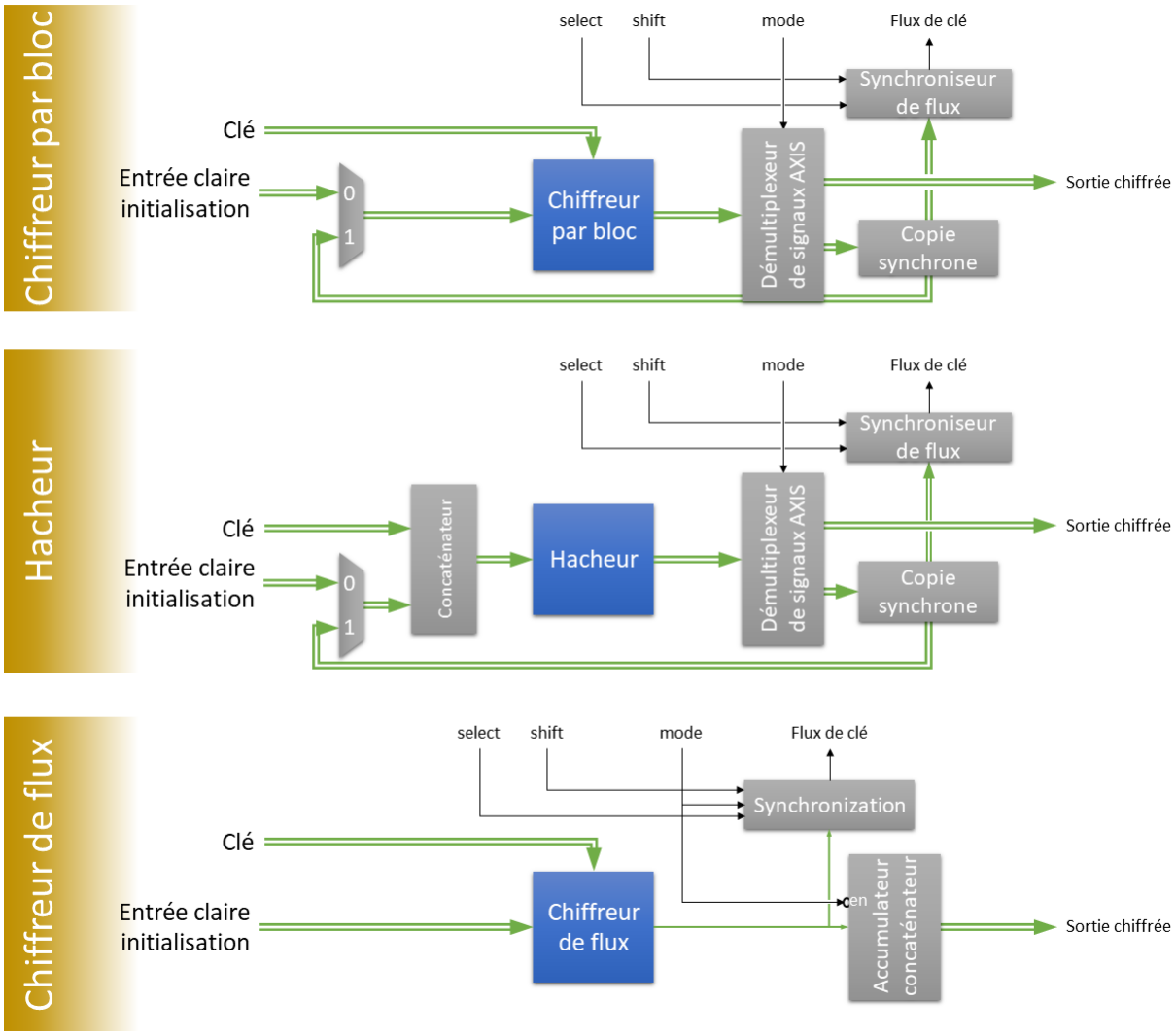


Figure II.16 Unité de chiffrement

II.7. Conclusion

Ce chapitre dédié à description du contrôleur SSAK a permis de voir quelles innovations ont pu être apportées pour satisfaire pleinement les exigences et les objectifs du projet tout en assurant un coût et un temps de connexion maitrisés. La politique de gestion des clés est la pierre angulaire de ce système ; elle permet la réduction du nombre de clés en mémoire ainsi que la diminution du temps d'exécution du protocole défi-réponse tout en évitant une clé unique et définitive pour tous les circuits.

Le protocole et les schémas d'implémentation physique présentés dans cette partie sont à la base des démonstrateurs de faisabilité et des implémentations d'évaluation qui sont décrits dans le Chapitre IV.

Enfin la dernière section a permis d'exposer une solution de sécurité complète de la chaîne de scan telle que demandée dans le projet HADES, avec une gestion d'accès dans l'infrastructure de test et une protection des données de test. Un autre aspect concernant la gestion au niveau industriel de cette sécurité complète est présenté dans le Chapitre III.

Chapitre III.
Test et chiffrement dynamique

Il existe dans la littérature de nombreuses propositions qui apportent des modifications ou des ajouts au standard IEEE1687 pour le sécuriser, pour augmenter son efficacité ou pour ajouter des fonctionnalités. Malheureusement, peu sont réellement utilisées dans l'industrie car il est souvent nécessaire de fournir un travail important pour adapter l'environnement de test (*test flow*) à ces nouveaux éléments. De plus, le risque de voir le nouveau système de test non fonctionnel, et de retarder la production de puces refroidit d'autant plus la volonté de porter les innovations sur des produits destinés à de très gros volumes. Les industriels étant assez conservateurs pour ces raisons, nous avons poussé notre travail pour automatiser l'authentification, et ainsi proposer une solution clé en main.

La première section de ce chapitre porte sur ce travail. La seconde partie présente une modification du système visant à étendre la confidentialité des vecteurs également à l'intérieur des circuits.

III.1. Automatisation de l'authentification

Cette section présente une étude ayant pour but d'intégrer l'authentification SSAK à une procédure de test automatisée. Les équipements de test automatisé (ATE) sont souvent utilisés pour effectuer les tests à travers les infrastructures de test incluant IEEE 1687. Comme expliqué dans la partie I.1.5, les vecteurs de test insérés dans les chaînes de scan sont normalement générés au préalable grâce à des descriptions du circuit, de la topologie du RSN, des procédures de test et des modèles de fautes ciblés. Cette génération hors-ligne rend difficile l'interaction avec le circuit car les vecteurs de test décidés à l'avance ne permettent pas à l'ATE de réagir aux différentes réponses du circuit. La première partie de cette section introduit le flot de test impliquant les ATE et la deuxième partie introduit MAST, l'environnement de CAO utilisé pour supporter le standard 1687, et ensuite montre comment l'authentification peut y être ajoutée et avec quelles limitations. La dernière partie explique comment intégrer MAST et le contrôleur SSAK pour pouvoir proposer une authentification automatisée.

III.1.1. Equipement de test et authentification

Les machines de test standard emploient des vecteurs de test générés au préalable. Un exemple de génération est donné par la Figure III.1 pour un flot basé sur IEEE 1687. Les différentes données de test, fournies par les instructions de test (PDL) pour chaque instrument, sont placées stratégiquement au sein des vecteurs de test pour que ceux-ci soient alignés avec les instruments dans la chaîne de scan. La position des instruments dans la chaîne de scan est fournie par un ou plusieurs fichiers ICL qui décrivent l'architecture du RSN. Cette technique réduit la charge de

travail des testeurs car aucune génération ne leur est demandé, et leur permet de travailler à leur vitesse maximale, ce qui est fondamental pour les tests de fabrications à haut volume. Cependant, à cause de ce rôle d'exécutants passifs ils ne sont pas en mesure de connaître l'état du système et d'interagir avec les circuits en cours de test : toute interprétation de résultats erronés est faite hors ligne.

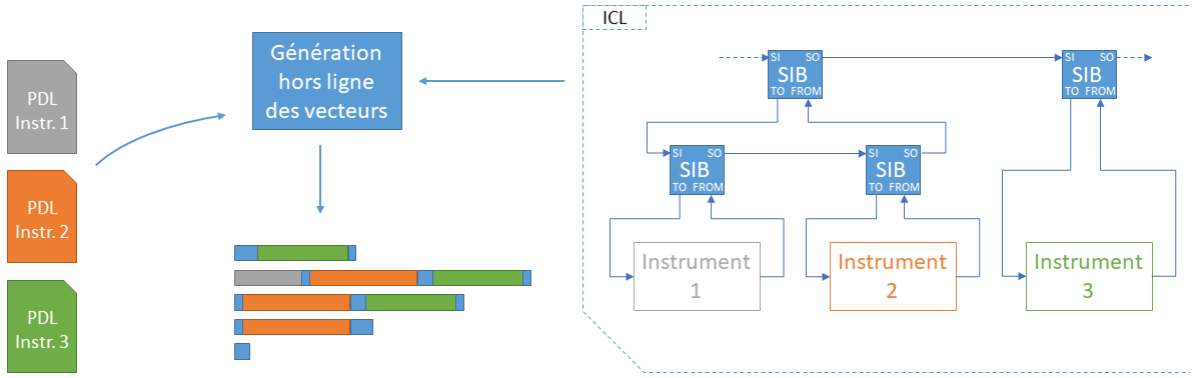


Figure III.1 Génération des vecteurs de test statiques

Il y a cependant une possibilité d'effectuer certaines actions interactives sur le circuit en programmant directement le testeur. Il est donc théoriquement possible d'effectuer une connexion sécurisée avec une méthode de test basée sur ATE, cependant cela présente trois limitations [46]. La première est la portabilité du programme ; le développement de l'algorithme serait propre à chaque système de test. La deuxième limitation vient de la difficulté à suivre l'état courant du système sous test. Ainsi, afin d'effectuer une authentification, le test doit être interrompu et le RSN mis en état initial ou dans un autre état connu. La troisième est la nature intrinsèquement déterministe d'une génération hors-ligne : toutes les données à échanger doivent être connues à l'avance, ce qui est en contraste avec une authentification de type défi-réponse.

Pour adresser ces problèmes, une solution de test pouvant interagir avec les circuits doit être employée. La partie suivante présente un environnement de test automatisé qui se présente comme une solution alternative.

III.1.2.Génération dynamique en ligne des vecteurs de test.

Dans [6] un environnement de test innovant est présenté. Il est notamment conçu pour pourvoir effectuer des tests où l'interaction est possible avec les instruments du standard IEEE 1687. Les stratégies de test standard cherchent à réduire au maximum la charge de travail des ATE pour que ceux-ci puissent tester le plus de circuits possibles le plus rapidement possible. Cette solution, *Manager for System-Centric Test* (MAST) vise à adresser des problématiques laissées ouvertes en proposant un flot complètement interactif, avec comme cible prioritaires :

- Gérer dynamiquement la topologie des RSN ;

- Faciliter la réutilisation des algorithmes topologiques ;
- Permettre l'exécution des algorithmes interactifs des instruments ;
- Offrir de riches capacités de débogage ;
- Accepter les interfaces autres que JTAG.

Pour garantir toutes ces fonctionnalité additionnelles, MAST ne peut pas partir de vecteurs de test niveau système (« top-level ») qui lui auraient été pré-générés. Comme représenté sur la Figure III.2, MAST a besoin de disposer entre autres des algorithmes de test des différents instruments et de la topologie du RSN. Grâce à ces connaissances il peut dynamiquement générer des vecteurs, qui peuvent configurer le RSN afin de permettre l'exécution des algorithmes d'instrument. En particulier, MAST est capable de tracer à tout moment l'état interne du Système, et il peut donc adapter sa stratégie de génération aux données qu'il reçoit. Du point de vue de ces instruments, l'environnement MAST permet de rendre transparente l'infrastructure de test ; l'exécution se déroule comme si les algorithmes étaient directement reliés aux instruments à l'intérieur du circuit. Il faut souligner que si les vecteurs de tests sont générés au niveau des IP, il est toujours possible de les encapsuler dans des procédures PDL et de laisser MAST se charger de la génération des vecteurs top-level. Cela garantit la rétrocompatibilité avec les approches d'ATPG actuelles.

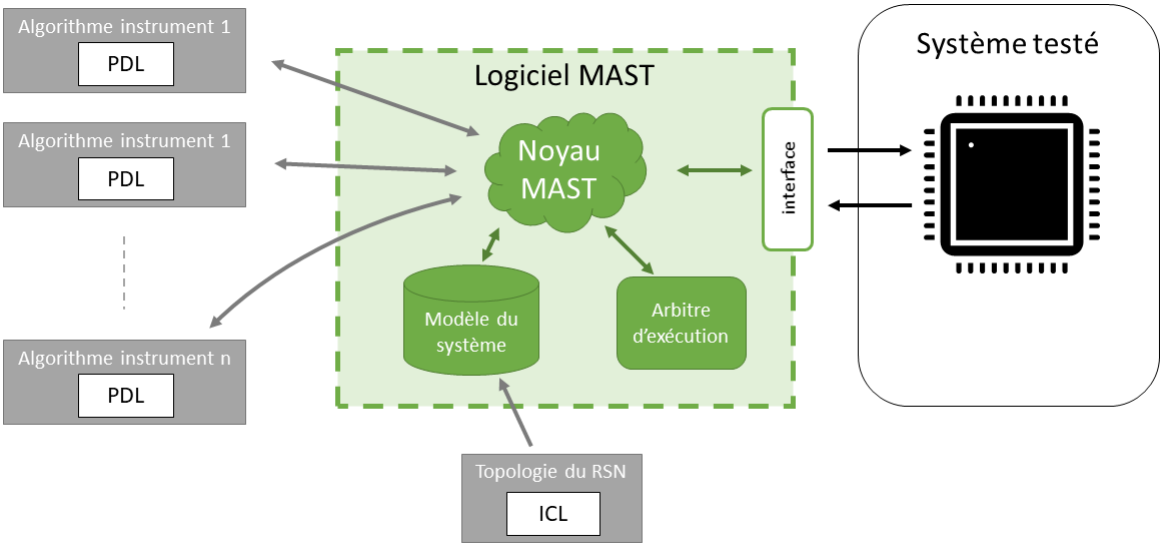


Figure III.2 Environnement de test MAST

MAST est développé en C++, il est donc possible de le porter sur la plupart des plateformes sans avoir à réécrire le code. En outre, il est capable de s'adapter à différentes interfaces pour se connecter à la chaîne de scan et n'a donc pas nécessairement besoin d'utiliser le contrôleur TAP pour effectuer les tests [47]. La Figure III.3, adaptée de [6], présente trois possibilités différentes d'utilisation et de portage de MAST. Le premier cas (a), permet d'utiliser MAST avec une version simulée du circuit testé (DUT), ce qui permet de développer et d'évaluer les procédures de test avant que le circuit physique ait été implémenté. Le schéma de

test (b) est une implémentation du système MAST sur le testeur industriel Advantest 93k, qui permet d'accéder à l'infrastructure du circuit testé. Enfin le cas (c) montre qu'il est possible de déployer MAST directement sur le circuit à tester si celui-ci est muni d'un processeur, comme par exemple un ARM ou un RISC V en ayant recours à de la cross-compilation.

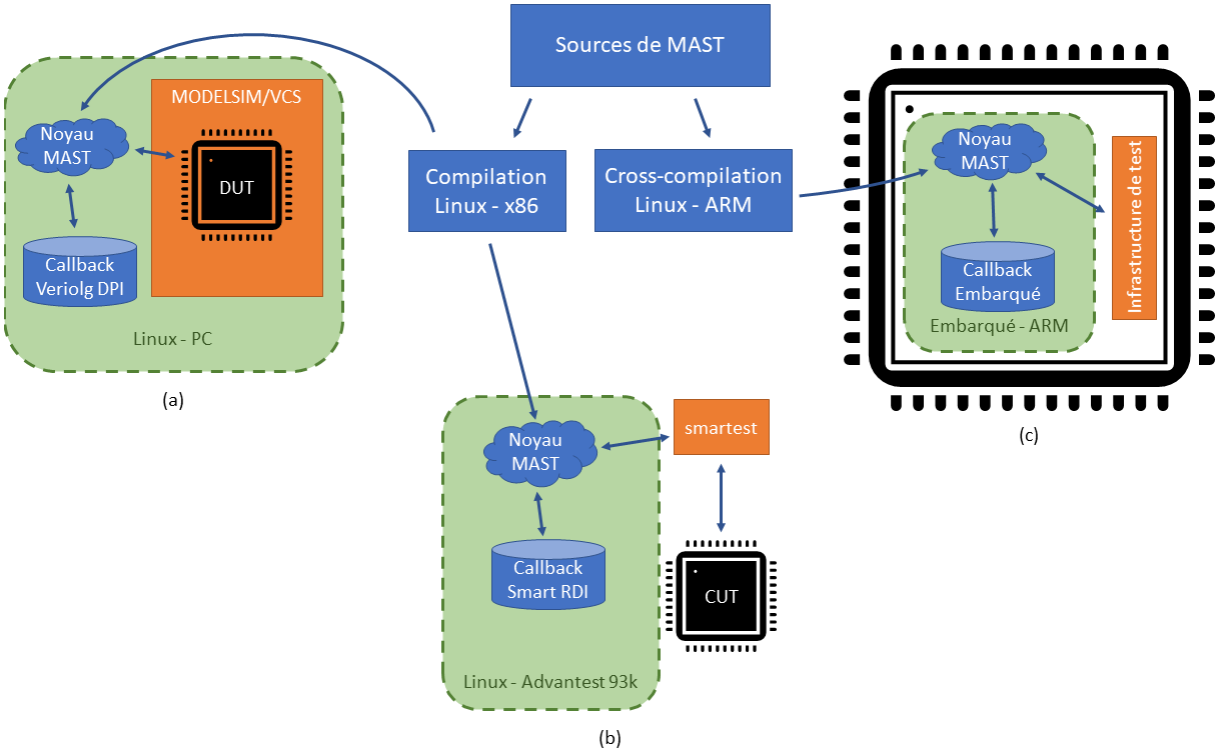


Figure III.3 Portabilité de MAST

III.1.3.Utilisation de MAST pour l'authentification SSAK

La partie précédente a permis de montrer que MAST possède les caractéristiques nécessaires pour pouvoir engager une authentification SSAK à un moment quelconque du test. La Figure III.4 montre les quatre acteurs principaux ainsi que les interactions nécessaires à une authentification SSAK automatisée par MAST. Préalablement à la phase de test, le fournisseur de sécurité doit intégrer l'implémentation matérielle du contrôleur au circuit testé et le plugin logiciel à l'hôte qui héberge MAST. En outre, il doit également fournir une clé secrète au circuit et un identifiant à l'utilisateur. Lors de l'exécution du test, si l'un des algorithmes de l'utilisateur doit interagir avec un segment protégé de la chaîne de scan, MAST prendra l'initiative de lancer l'authentification SSAK pour pouvoir garantir l'accès de manière transparente.

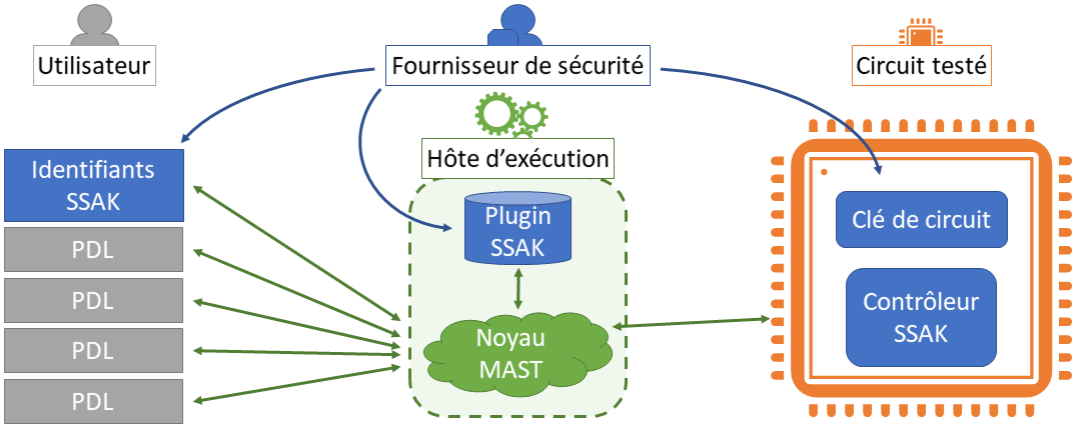


Figure III.4 Authentification SSAK automatisée par le système MAST

Pour rendre maximum son adaptabilité et sa flexibilité le plugin tout comme MAST suit un développement Orienté Objet. La Figure III.5 est une partie du diagramme de classe des objets qu'utilise MAST pour modéliser les RSN. La partie mise en emphase par la couleur orange sur la Figure III.5 correspond à la partie d'authentification, développée pour le protocole SSAK alors que le reste est une partie de MAST. La classe abstraite *Selectable* est utilisée par MAST pour représenter dans son modèle les éléments de la chaîne de scan pouvant reconfigurer le RSN. Par exemple les ScanMux sont implémentés dans MAST par une classe fille de *Selectable*. La classe principale concernant le SSAK est *SSAKplugin*. Elle hérite également de la classe *Selectable* qui contient notamment trois méthodes :

- *select()* ou fonction d'activation, est utilisée pour activer l'objet représenté ce qui peut représenter différentes actions en fonction du type de cet élément,
- *deselect()* ou fonction de désactivation, est utilisée pour désactiver l'élément piloté,
- *isSelected()* ou fonction de vérification, sert à interroger l'état actif ou inactif de l'élément.

Pour reprendre l'exemple du SIB, la fonction d'activation ouvre le SIB en inscrivant un « 1 » dans son registre d'état, la fonction de désactivation ferme le SIB en y inscrivant un « 0 » et la fonction de vérification retourne l'état actuel du SIB.

Cette approche est basée sur l'utilisation des « fonctions de rappel » (« callback » en anglais), une technique informatique classique pour permettre l'ajout de fonctionnalités à un logiciel existant, et qui est notamment proposée par l'extension P1687.1 du standard 1687 [48].

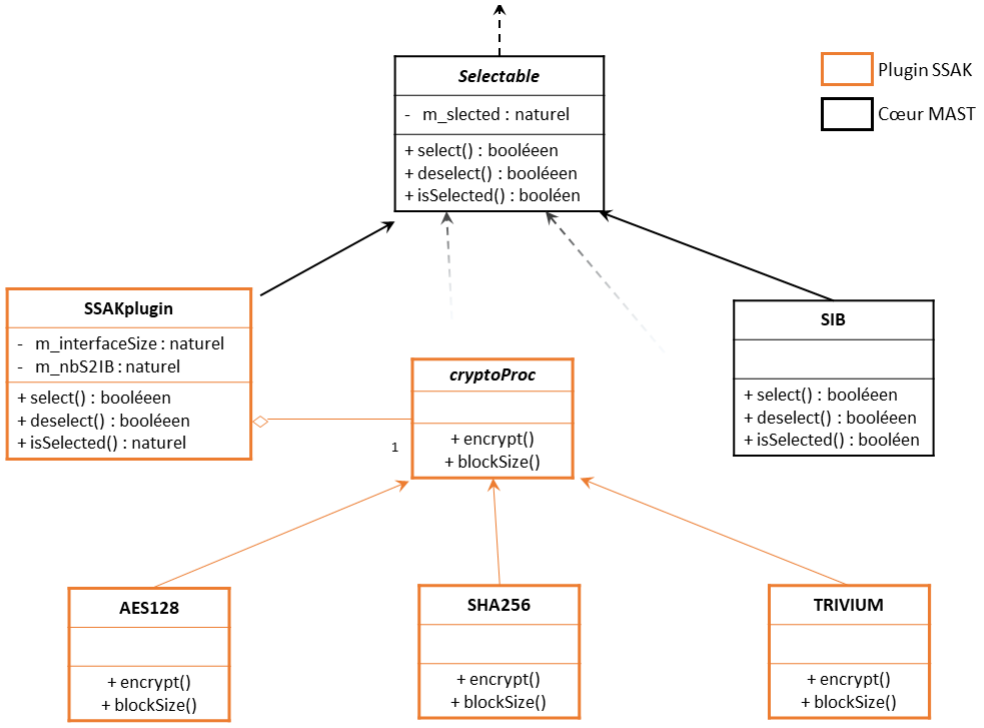


Figure III.5 Diagramme de classe du plugin SSAK

Pour « SSAKplugin » l'activation consiste à faire une authentification ; pour cela le plugin doit disposer d'un cryptoprocesseur. Celui-ci est présent dans une classe abstraite nommée « *cryptoProc* » dont peuvent hériter les classes de différents éléments cryptographiques, pouvant être des chiffreurs par bloc, des hacheurs et des chiffreurs de flux. Deux méthodes virtuelles pures de la classe « *cryptoProc* » ressortent du schéma : la fonction « *blockSize*() » qui retourne la taille des messages qui sont utilisés, ce qui correspond également à la taille du segment d'interface du contrôleur sur la chaîne de scan, et la fonction « *encrypt*() » qui permet de configurer et d'utiliser le cryptoprocesseur en fonction de son type. L'utilisation des données dans les différents types de cryptoprocesseurs, illustrée par la Figure III.6, est similaire à leur usage dans l'implémentation matérielle du contrôleur décrite dans la partie II.3.2 :

- Chiffreurs par bloc (a) ; le défi est utilisé comme texte clair, la SSAK comme clé, le résultat du chiffrement donne la réponse du défi.
- Hacheurs (b) : le défi et la SSAK sont concaténés pour former un seul vecteur, en veillant à ce que l'ordre des bits soit identique à celui du hacheur implémenté dans le circuit, la réponse est obtenue une fois ce vecteur haché.
- Chiffreurs de flux (c) : le défi est utilisé comme vecteur d'initialisation, la SSAK est la clé, et le flux de clé est utilisé pour former bit à bit la réponse du défi, jusqu'à ce que la taille requise soit atteinte.

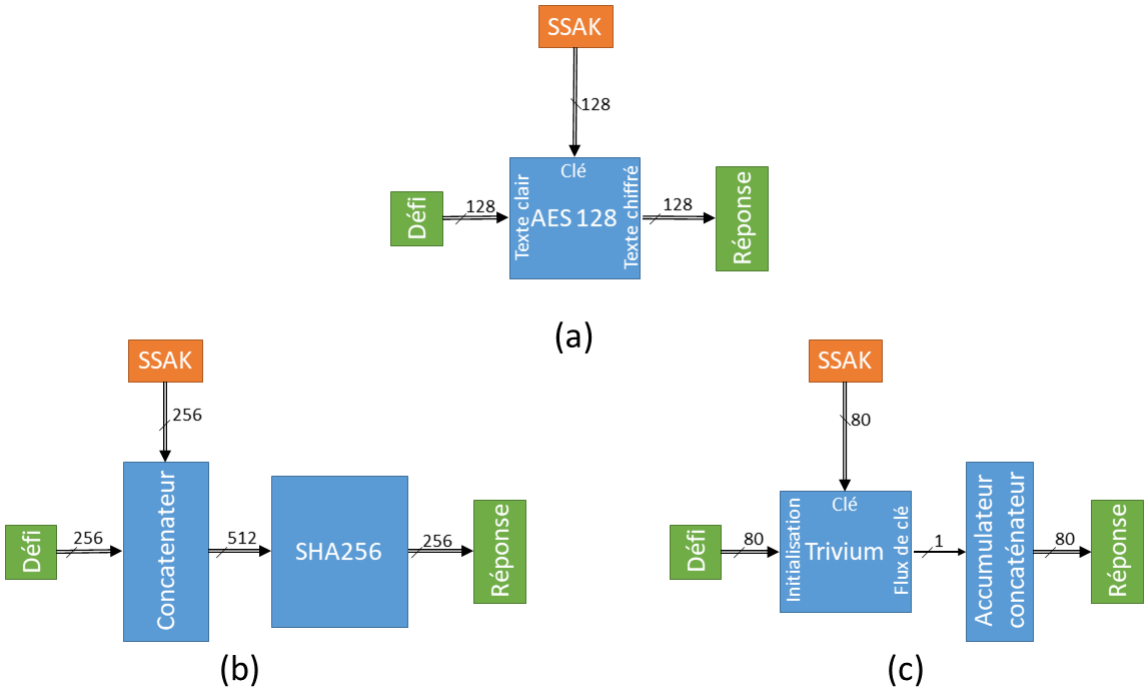


Figure III.6 Utilisation de différents cryptoprocesseurs dans MAST

L'utilisation de classes et du polymorphisme dans le plugin permet d'une part d'intégrer rapidement le plugin à MAST et de faciliter sa maintenance ; d'autre part, cette approche permet le changement ou l'intégration des nouveaux cryptoprocesseurs rapidement. En effet, il n'est pas nécessaire de modifier le code du plugin existant, la définition d'une nouvelle classe héritant de *cryptoProc* suffit.

Afin de rendre visible les différences de comportement que peuvent avoir les tests avec authentification utilisant MAST et ceux utilisant une authentification par PDL, un scénario de test a été élaboré. La Figure III.7 représente ce scénario dans le cas où MAST est utilisé. Le circuit de test piloté par MAST comprend quatre SIB. Le premier dans l'ordre du circuit abrite le contrôleur SSAK, le second et le troisième sont des SIB sécurisés (S²IB) et contiennent respectivement l'instrument 1 et 2, enfin le dernier SIB contient l'instrument 3 qui n'est donc pas protégé.

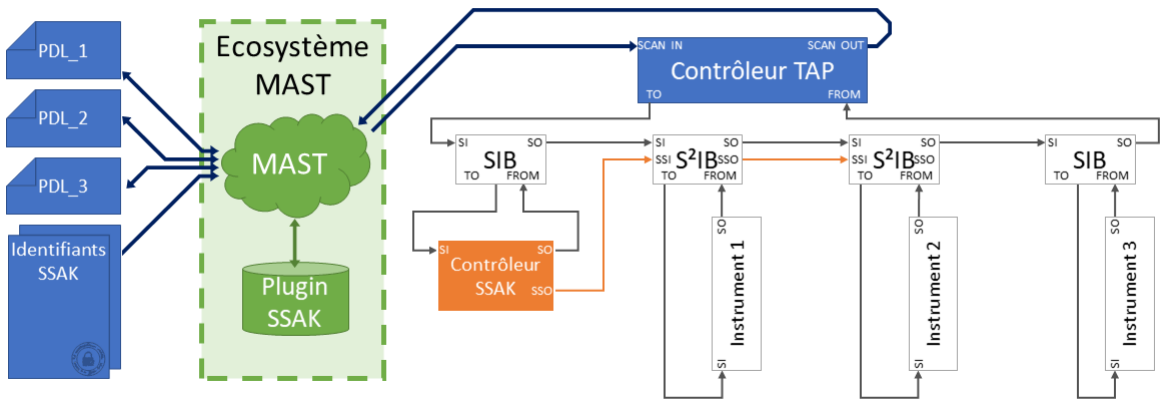


Figure III.7 Exemple d'architecture incluant MAST et SSAK

La Figure III.8 montre les comportements attendus lors d'un test devant procéder à une authentification MAST. Dans ce scénario trois instruments doivent être testés, sachant que l'instrument contrôlé par le PDL 1 ainsi que celui contrôlé par le PDL 2 sont protégés grâce à SSAK. L'instrument contrôlé par le PDL_3 ne requiert pas un accès protégé. Lorsque l'authentification est effectuée dans le code des PDL (a), les authentifications de segment ne peuvent pas avoir lieu pendant qu'un autre test est effectué. En outre, supposition est faite que chaque PDL est uniquement en possession des identifiants permettant l'accès à l'instrument qu'il est sensé tester, ce qui empêche d'effectuer les test PDL 1 et 2 en même temps. A l'opposé, lorsque l'authentification est prise en charge par MAST (b), il est possible d'effectuer l'authentification pendant le test d'autres parties du circuit, ce qui permet du parallélisme. De plus MAST gérant l'intégralité du système, il est possible de lui fournir des identifiants couvrant toutes les parties protégées du système qui doivent être testées, ici en l'occurrence l'instrument du PDL 1 et celui du PDL 2. L'échelle de temps en abscisse (unité de temps arbitraire) permet d'avoir un aperçu, et un point de comparaison pertinent entre les deux méthodes. Il est possible de voir ici que l'utilisation de MAST pour l'authentification a réduit considérablement le temps de test. Toujours avec MAST mais dans le cas où l'utilisateur ne possède pas les droits d'accès à tous les instruments, en l'occurrence dans l'exemple (c) où il manque l'accès à l'instrument 2, les tests des autres segments pourront se dérouler sans être perturbés.

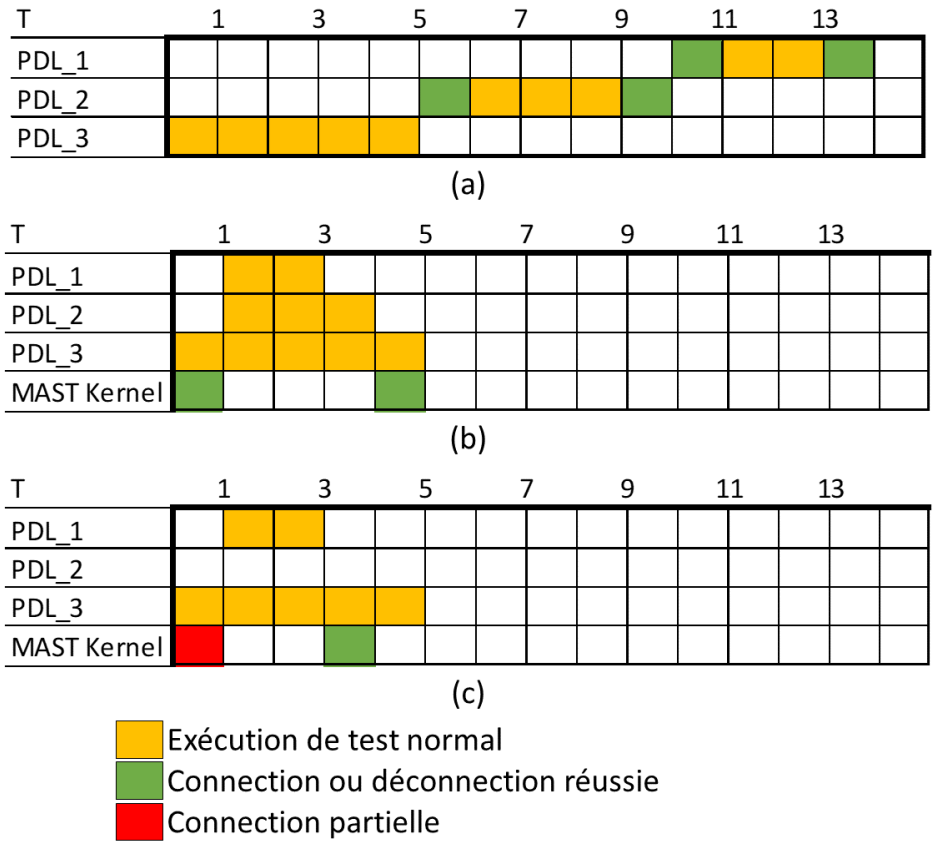


Figure III.8 Authentification par PDL (a), authentification par MAST (b), authentification partielle avec MAST (c). Seuls PDL 1 et 2 requièrent une authentification.

III.2. Chiffrement interne

Comme évoqué précédemment lors des sections II.1 et II.6, la gestion des accès n'est pas à elle seule suffisante pour garantir une sécurité du circuit ; il est préférable de garantir également la confidentialité des vecteurs de test. Afin d'atteindre ce second objectif, il est possible d'avoir recours à du chiffrement de vecteur dans la chaîne de scan [45]. La section II.6 propose une solution capable de garantir une confidentialité des données en dehors de la chaîne de scan du circuit. La solution présentée ici vise à étendre cette confidentialité aussi à l'intérieur de la chaîne de scan.

III.2.1. Confidentialité des données hors de leurs segments

Pour fournir une solution de confidentialité adaptée, il faut connaître le profil des éventuels espions. La Figure III.9 présente deux modèles d'attaquants. Les techniques de défense évoquées jusqu'ici sont correctement efficaces face à la situation (a), où l'espion se trouve à l'extérieur du circuit qui doit être protégé. C'est le modèle d'attaque auquel la technique présentée en II.6 cherche à répondre, c'est pourquoi les données de test sont chiffrées uniquement en dehors du circuit. Le modèle (b), celui qui nous intéresse ici, prévoie que la fuite de données puisse avoir lieu à l'intérieur du circuit soit à cause d'un cheval de Troie soit à cause d'un instrument d'un tiers indigne de confiance. Ainsi dans ce scénario le chiffrement en sortie de circuit et le déchiffrement en entrée ne suffisent plus à assurer la confidentialité des données, il faut également que les vecteurs soient chiffrés à l'intérieur du circuit.

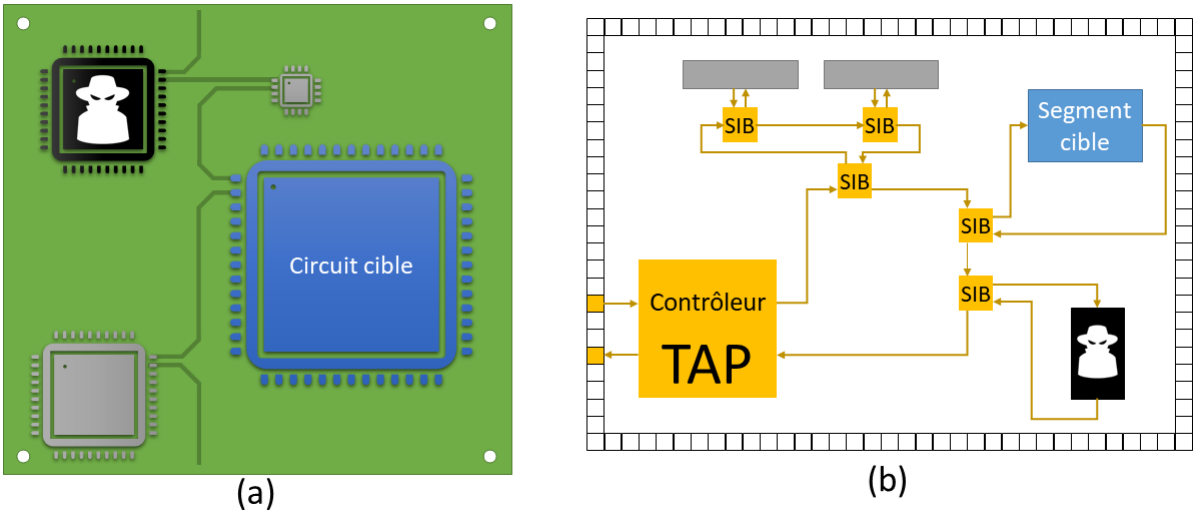


Figure III.9 Modèles d'attaquant : (a) externe, (b) interne

Pour assurer une confidentialité complète des données dans le circuit, il faudrait que les vecteurs soient déchiffrés à l'entrée de chaque segment et que les réponses soient chiffrées à la sortie du segment, ce qui en fonction des circuits, peut représenter une grande complexité. Afin de rendre l'implantation plus raisonnable, le choix est fait d'effectuer ces chiffrements et déchiffrements uniquement pour les

segments qui ont besoin d'être protégés. Ces segments protégés sont les mêmes que ceux utilisant la protection SSAK. La Figure III.10 représente l'exemple d'une approche théorique pour apporter la confidentialité des données des segments protégés. Comme illustré, la solution implique d'effectuer des chiffrements et des déchiffrements distribués au long de la chaîne de scan. Dans ce cas, utiliser des chiffreurs pour chaque segment sécurisé engendrerait des coûts non envisageables pour une infrastructure de test. Une solution crédible à ce problème doit donc avant tout trouver le moyen de baisser considérablement les coûts de cette pratique.

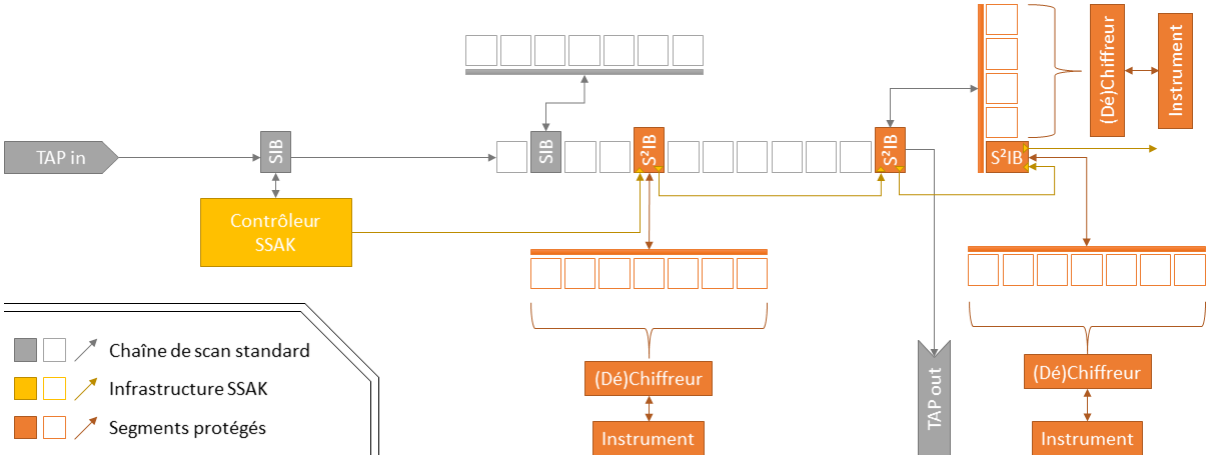


Figure III.10 Approche théorique de la protection des données de segments à l'intérieur du circuit

III.2.2.Chiffrement par SIB

Pour réduire au maximum l'introduction de nouveau matériel dans le circuit, il a été choisi de réutiliser celui introduit par la solution SSAK, notamment la chaîne de scan sécurisée ainsi que les S²IB, et la solution de chiffrement développée en fin de Chapitre II. Dans cette technique visible sur la Figure II.15, un seul cryptoprocesseur est chargé de générer un flux de clé utilisé à deux emplacements de la chaîne de scan. Afin de procéder à la protection des données en interne, cette même méthode est réemployée. Il est possible d'effectuer ces chiffrements distribués au sein des S²IB parce que chacun des segments protégés est raccordé à l'infrastructure de test par un S²IB. Ainsi, ces derniers ont été modifiés pour être capables d'effectuer un déchiffrement en amont et un chiffrement en aval du segment protégé. Cette évolution des S²IB est nommée « *encryption Segment Insertion Bit* » (eSIB). Comme le montre la Figure III.11, un ajout très simple de matériel est nécessaire pour cela : en effet, seules deux portes « ou exclusif » mono-bit sont requises.

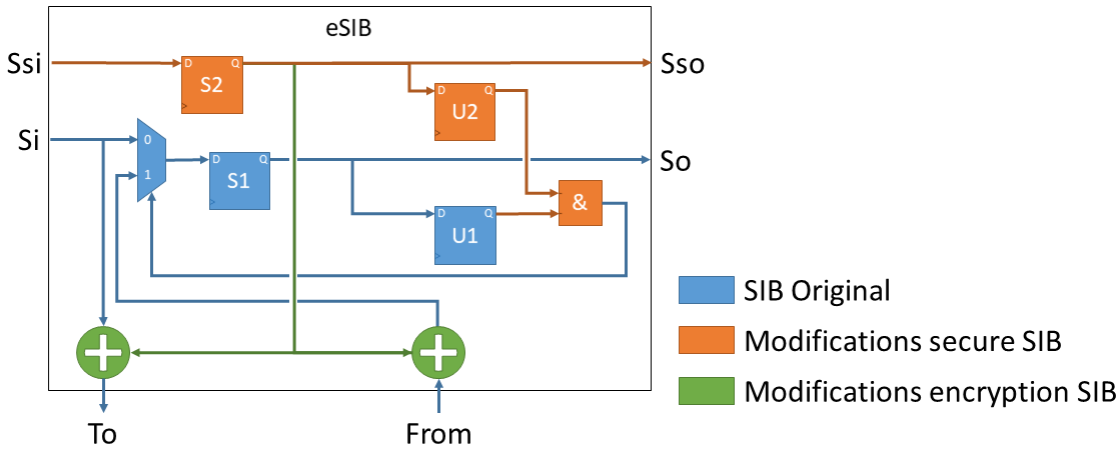


Figure III.11 eSIB : Ajout d'une fonction de chiffrement au S²IB

L'utilisation de eSIB permet également de réutiliser la chaîne de scan sécurisée (SSC) pour transmettre la clé de flux à tous les chiffreurs ainsi ajoutés. Le flux de clé proposé par le générateur déjà utilisé et présenté en partie II.6.2 peut être employé. Un arbitre doit être placé en tête de chaîne de scan sécurisée pour choisir entre les données venant du contrôleur SSAK et celles venant du générateur. Par construction le chiffrement n'est fonctionnel que lorsqu'un segment sécurisé est actif, mais pour garantir que l'autorisation d'accès et le chiffrement de chaîne n'utilisent pas la SSC en même temps, le chiffrement n'est activé qu'une fois l'authentification SSAK effectuée. La Figure III.12 montre les changements qu'apporte cette solution au circuit ; ils sont très marginaux en termes de coût par rapport à ce que semblait exiger l'approche initiale proposée en Figure III.10.

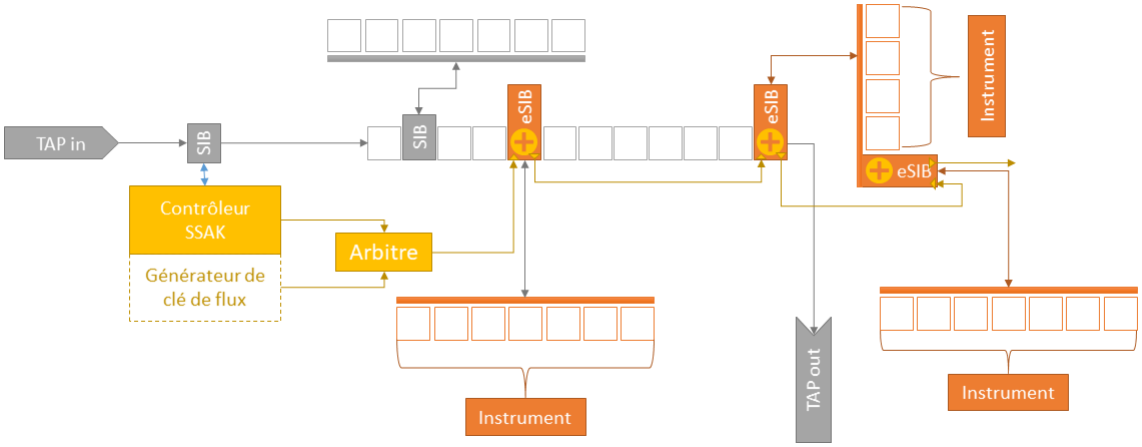


Figure III.12 Propagation de la clé de flux dans la chaîne de scan sécurisée

III.2.3.Gestion des déphasages de clés et contrôle

La solution matérielle présentée jusqu'ici laisse de par sa simplicité un problème ouvert : la position des portes *XOR* de chiffrement au milieu de la chaîne de scan provoque un déphasage entre les données et la clé, de plus la reconfiguration du RSN peut modifier ce déphasage et ajouter ou supprimer des points de chiffrement dans la chaîne de scan. L'utilisation d'un chiffrement par flux permet de corriger ce décalage simplement en appliquant un délai ou une avance

à la clé de flux appliquée par l'utilisateur. La Figure III.13 montre le schéma du cas général de cette correction de décalage pour l'écriture et la lecture dans la chaîne de scan avec un seul nœud de chiffrement.

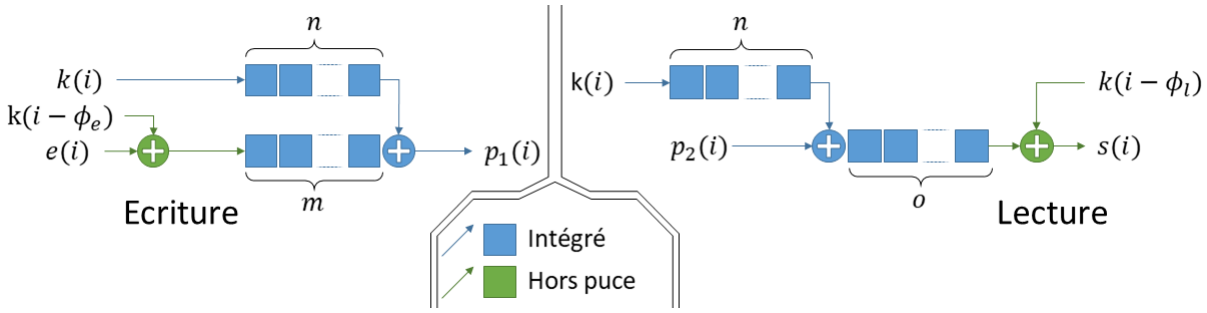


Figure III.13 Schéma du cas général du chiffrement intene pour l'écriture et la lecture

Avec :

- k la clé de flux,
- ϕ_e et ϕ_l respectivement le déphasage de clé d'écriture et de lecture,
- e le vecteur d'entrée avant chiffrement,
- n la longueur de la chaîne de scan sécurisée,
- m la longueur de la chaîne de scan avant le chiffreur intégré,
- o la longueur de la chaîne de scan après le chiffreur intégré,
- p_1 le vecteur au début du segment protégé,
- p_2 le vecteur à la fin du segment protégé,
- s le vecteur récupéré par l'utilisateur une fois déchiffré.

Pour le cas de l'écriture, les données déchiffrées dans le circuit peuvent être exprimées ainsi :

$$p_1(i) = e(i-m) \oplus k(i-\phi_e-m) \oplus k(i-n)$$

Pour que le vecteur soit déchiffré correctement, il faut que le vecteur avant chiffrement $e(i)$ corresponde au vecteur déchiffré $p_1(i)$ mais décalé de m cycles, soit : $p_1(i) = e(i-m)$. A partir de ces deux égalités, il est possible de déduire le déphasage ϕ_e nécessaire :

$$\begin{aligned} & e(i-m) = e(i-m) \oplus k(i-\phi_e-m) \oplus k(i-n) \\ & \Leftrightarrow k(i-n) = k(i-\phi_e-m) \\ & \Rightarrow i-n = i-\phi_e-m \\ & \Rightarrow \boldsymbol{\phi_e = n-m} \end{aligned}$$

Il faut donc pour l'écriture retarder la clé de flux de la longueur de la chaîne de scan avant cette porte XOR moins la longueur de la SSC avant celle-ci. Concernant la lecture la même méthodologie peut s'appliquer :

$$s(i) = p_2(i-o) \oplus k(i-n-o) \oplus k(i-\phi_l)$$

Avec $s(i) = p_2(i - o)$ pour avoir un chiffrement et un déchiffrement cohérant :

$$\Rightarrow k(i - n - o) = k(i - \phi_l)$$
$$\Rightarrow i - \phi_l = i - n - o$$
$$\Rightarrow \boldsymbol{\phi_l = n + o}$$

Pour l'écriture il faut donc avancer la clé de la somme de la longueur de scan après la porte XOR intégrée et de la longueur de la SSC jusqu'à l'emplacement de cette même porte.

Ces équations permettent de décrire simplement les déphasages à appliquer aux clés de flux par l'utilisateur dans le cas d'un unique chiffrement dans l'infrastructure de test. Cependant, le chiffrement interne ne reposera pas uniquement sur un chiffreur intégré au milieu de la chaîne : l'architecture prévoit un chiffrement à chaque extrémité de chaque segment protégé. La Figure III.14 montre un exemple avec quatre chiffrements internes. Heureusement grâce aux propriétés de l'opération ou exclusif, il est possible d'interchanger l'ordre des opérations sans que le résultat final ne soit impacté.

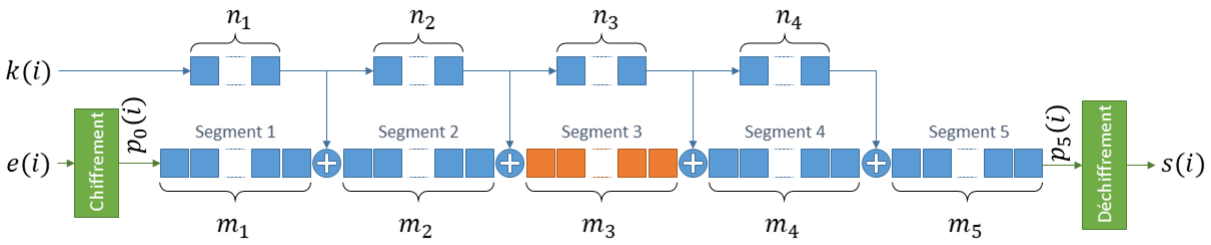


Figure III.14 Chaîne de scan interne avec plusieurs chiffrement internes

L'exemple de la Figure III.14 s'intéresse à la protection des données du segment 3. Deux chiffrements doivent être appliqués au vecteur d'entrée pour s'adapter aux deux chiffrements internes appliqués avant le segment 3. En reprenant les équations de déphasage pour l'écriture on obtient les deux décalages suivants :

$$\phi_{e_1} = n_1 - m_1$$
$$\phi_{e_2} = n_1 + n_2 - m_1 - m_2$$

Le chiffrement pour le segment 3 peut donc être exprimé ainsi :

$$p_0(i) = e(i) \oplus k\left(i - \phi_{e_1}\right) \oplus k(i - \phi_{e_2})$$

Concernant la lecture des vecteurs de tests, deux XOR de clé déphasés sont aussi nécessaires :

$$\phi_{l_1} = n_1 + n_2 + n_3 + m_4 + m_5$$
$$\phi_{l_2} = n_1 + n_2 + n_3 + n_4 + m_5$$

L'équation de chiffrement est donc :

$$s(i) = p_5(i) \oplus k\left(i - \phi_{l_1}\right) \oplus k(i - \phi_{l_2})$$

Malgré la complexité induite par un nombre de chiffrements effectués à l'intérieur de la chaîne de scan potentiellement élevé, ces équations permettent d'avoir un chiffrement et un déchiffrement efficaces à l'extérieur du circuit par l'utilisateur. Cependant, les longueurs de chaînes de scan exprimées dans les schémas ainsi que dans les équations ne sont pas nécessairement des constantes ; en effet, la reconfiguration de la chaîne de scan entre les cycles amène à des variations de longueur au sein de l'infrastructure de test. Pour être capable d'utiliser cette technique de chiffrement il faut que l'utilisateur soit capable de connaître à tout moment la longueur effective de la chaîne de scan, afin d'adapter les déphasages des clés de chiffrement.

La section III.1 rappelle qu'il est difficile d'utiliser des ATPG standards pour effectuer des tests interactifs ; leur utilisation ici devient encore davantage compliquée. Cependant, l'outil MAST gère un modèle qu'il est capable de faire évoluer en temps réel pour correspondre à l'architecture du RSN au moment présent. Les informations issues de son modèle et l'utilisation des équations présentées ici permettent donc de pouvoir appliquer cette technique de chiffrement en calculant les bons déphasages de clé de flux en temps réel pour garantir la confidentialité des données entre l'utilisateur et les segments protégés.

III.3. Conclusion

Dans ce chapitre, une solution clé en main a été proposée pour adapter l'utilisation de l'authentification et du chiffrement aux environnements de test industriels. Pour promouvoir au mieux SSAK et le rendre plus accessible, il est nécessaire de faciliter au maximum son intégration au flot de test. C'est pour aller dans ce sens que le développement d'une solution de connexion transparente et automatisée s'appuyant sur MAST a été entrepris. De plus ce développement apporte également des bénéfices en performance car contrairement à une solution basée sur des ATE, il n'est nul besoin d'interrompre des tests de segments en cours pour procéder à une authentification.

Dans le but d'étendre la confidentialité des données lors de leur transfert dans la chaîne de scan interne, le chiffrement par eSIB est introduit. Les très faibles modifications à apporter au système SSAK pour son intégration présentent un avantage certain. En revanche la difficulté de mise en place des algorithmes utilisateur présenterait un problème difficilement surmontable s'il n'était pas possible de l'automatiser en employant MAST. Bien qu'en théorie cette solution de protection des données semble peu coûteuse et raisonnablement facile à déployer il est tout de même nécessaire de produire un prototype pour prouver sa faisabilité et étudier les coûts réels d'implémentation. Ces deux tâches sont présentées dans le Chapitre IV.

Chapitre IV.
Démonstrateurs et Résultats

Les chapitres précédents décrivent les différentes composantes d'une solution voulue pour sécuriser l'accès à des parties ciblées de la chaîne de scan interne d'un circuit. S'il est capital de sécuriser les RSN au moyen de fonctionnalités comme l'authentification ou le chiffrement des données, il reste important de garder en mémoire qu'il s'agit de fonctionnalités de test et qu'elles doivent par conséquent être limitées en surface d'implémentation ainsi qu'en temps d'exécution.

Ce chapitre à deux objectifs : il doit montrer qu'il est possible d'implémenter ces solutions de sécurité mais aussi qu'elles sont viables et performantes. C'est pourquoi le chapitre sera divisé en plusieurs sections. La première se focalise sur les démonstrateurs qui ont pour but de démontrer la faisabilité des systèmes. Considérant que la grande majorité des coûts d'implémentation de ces systèmes vient du crypto processeur choisi, la seconde section s'attellera à comparer leurs prix et leurs performances. La dernière section enfin cherchera à comparer les différents systèmes avec ceux de la littérature déjà mentionnés au Chapitre I.

IV.1. Démonstrateurs

Durant les trois années, des démonstrateurs ont été produits pour prouver de manière pratique et incrémentale la faisabilité de la proposition SSAK, puis de ses différentes modifications et options. Les démonstrateurs reposent sur des FPGA où un RSN est implémenté, et sécurisé avec le système dont la faisabilité veut être étudiée. Il existe ainsi quatre types de démonstrateurs : le premier implémente uniquement le contrôleur SSAK, le deuxième est un démonstrateur pour le projet HADES, qui implémente le chiffrement de chaîne de scan ainsi que l'authentification, le troisième montre la possibilité et la pertinence d'utiliser MAST avec la solution proposée et enfin le dernier est une implémentation préliminaire du chiffrement interne des vecteurs de test.

Cette section va présenter à tour de rôle ces différents démonstrateurs et apporter un œil critique sur l'intérêt et la facilité de leur utilisation.

IV.1.1. Authentification seule

Ce premier démonstrateur a pour but de démontrer la faisabilité du système SSAK. Il met en place dans un FPGA Zynq 7000 une chaîne de scan reconfigurable précédée du contrôleur d'authentification. En plus de la partie logique programmable, ce système sur puce (SoC) possède comme le montre la Figure IV.1 un processeur ARM intégré. Ce processeur est utilisé pour contrôler la chaîne de scan interne implémentée, grâce à l'utilisation de périphériques AXI.

Une application exécutée par le possesseur ARM sert d'interface entre l'utilisateur et la chaîne de scan. Elle assure également la fonction d'authentification SSAK coté utilisateur. L'utilisateur doit fournir ses identifiants pour que l'application puisse déverrouiller les S²IB. Il est ensuite possible d'écrire et de lire des valeurs dans les segments protégés. Pour vérifier que les valeurs écrites dans le premier segment le soient effectivement, il est possible de les relire grâce au deuxième segment. Quant à eux, les vecteurs écrits dans le deuxième segment peuvent être vérifiés au moyen des LEDs présentes sur la carte Zybo.

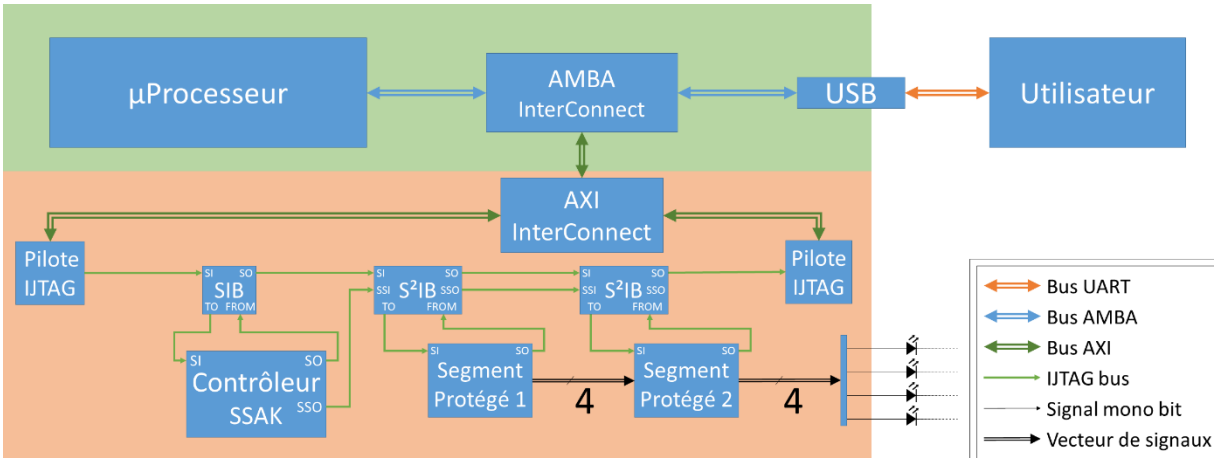


Figure IV.1 Architecture générale du premier démonstrateur

Le fonctionnement probant de ce démonstrateur a permis de déclarer la solution comme viable et fonctionnelle et ainsi pouvoir l'exposer à la communauté scientifique lors de l'*International Verification and Security Workshop* (IVSW) en 2019 [49].

IV.1.2. Authentification et chiffrement

La solution décrite dans la section II.6 a pour objet de rassembler et de fusionner la technique de chiffrement des vecteurs [43] et la technique de gestion des accès SSAK. Cette solution composée a été sélectionnée pour répondre aux objectifs de HADES concernant le test sécurisé. Pour vérifier la faisabilité de cette technique, un premier démonstrateur a été implémenté sur le même FPGA que celui cité dans la section précédente. Il fut l'objet d'une exhibition lors de l'évaluation du laboratoire TIMA par le Haut Conseil de l'évaluation de la recherche et de l'enseignement supérieur (HCERES) pendant laquelle il fit preuve de son bon fonctionnement.

La présence de l'entreprise JTAG dans le consortium du projet HADES a aussi permis de bénéficier d'une carte de développement spécialement conçue pour éprouver des infrastructures de test personnalisées. La structure de ce démonstrateur est représentée dans la Figure IV.2, la représentation ne montrant qu'une partie de ce qui est présent sur la carte. Il est tout de même possible de voir que cette carte implémente deux réseaux de test JTAG. Le premier, connecté au « *port JTAG* », est la chaîne de scan standard de la carte prévue pour tester les

différents circuits ainsi que pour programmer le FPGA, c'est d'ailleurs par cette chaîne que la description matérielle du démonstrateur est implémentée. Le second réseau, connecté au « *port JTAG logiciel* », est une chaîne reliée à des entrées et sorties standard du FPGA, ce qui permet d'implémenter un contrôleur TAP et de le relier directement à ce circuit.

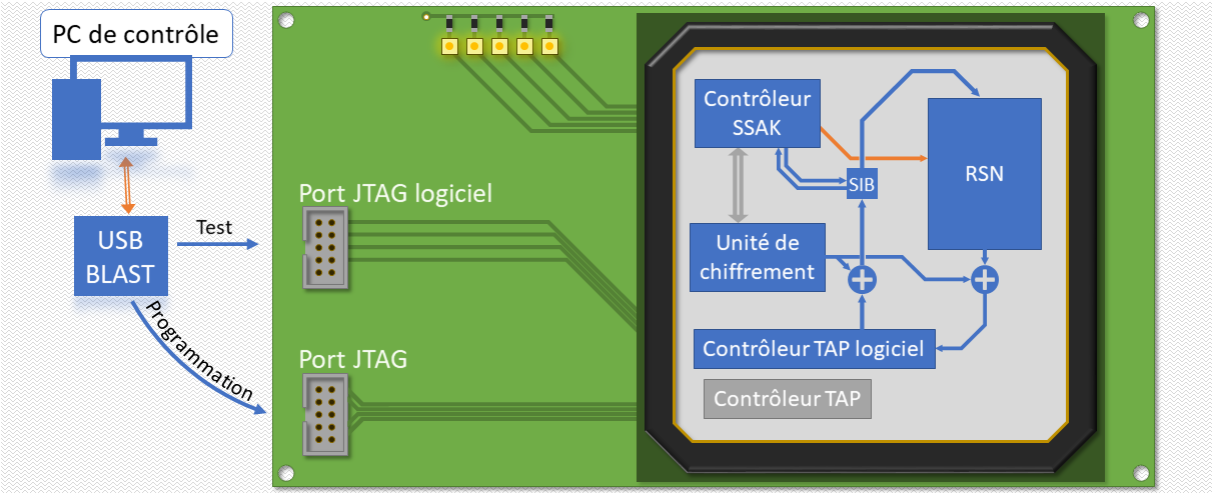


Figure IV.2 Architecture du démonstrateur SSAK et chiffrements fusionnés sur carte JTAG

Pour rendre visibles les états des S²IB à l'intérieur du RSN, des LED trois couleurs Rouge Vert Bleu (RVB) présentes sur la carte de développement sont reliées aux SIB pour suivre leur état. Chacune est reliée à un S²IB, et le code couleur de la Figure IV.3 permet de connaître son état :

- Rouge : l'accès au segment protégé par le S²IB n'est pas autorisé.
- Vert : le S²IB est ouvert, les sous segments sont donc accessibles.
- Bleu : reflète l'état du signal select, cela signifie en plus que le segment dans lequel se trouve le S²IB est activé.

A l'initialisation, les LED sont rouges car aucun accès n'est autorisé donc les S²IB sont fermés et non sélectionnés. Les LED peuvent ensuite afficher différentes combinaisons de couleur en fonction de l'état de leur S²IB respectif. Cependant, elles ne peuvent être ni jaunes ni blanches, car il est impossible qu'un S²IB soit ouvert alors que l'autorisation d'ouverture n'est pas accordée. L'utilisation de ces LED permet à la démonstration d'avoir un contenu visuel qui reflète l'état du système ainsi que les autorisations de manière immédiate.

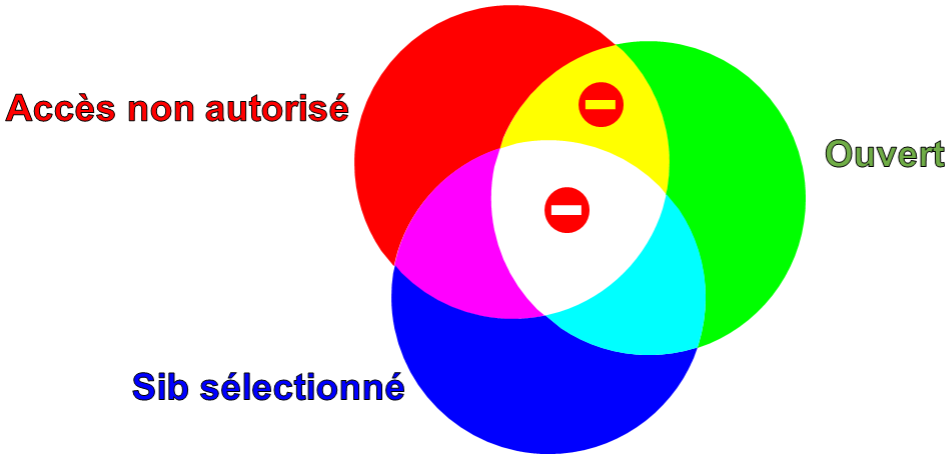


Figure IV.3 Code couleur des LED du démonstrateur authentification et chiffrement

Afin de juger de la pertinence de la fusion des systèmes d'authentification et de chiffrement en ce qui concerne les coûts en ressources, d'autres versions du démonstrateur ont été implémentées, comme illustré en Figure IV.4 :

- (a) Aucune technique de protection n'est implémentée,
- (b) La protection se fait uniquement grâce à la méthode SSAK,
- (c) Implémentation du chiffrement des données uniquement,
- (d) Le RSN est protégé par les deux systèmes implémentés côte à côte, sans mutualisation de ressources.

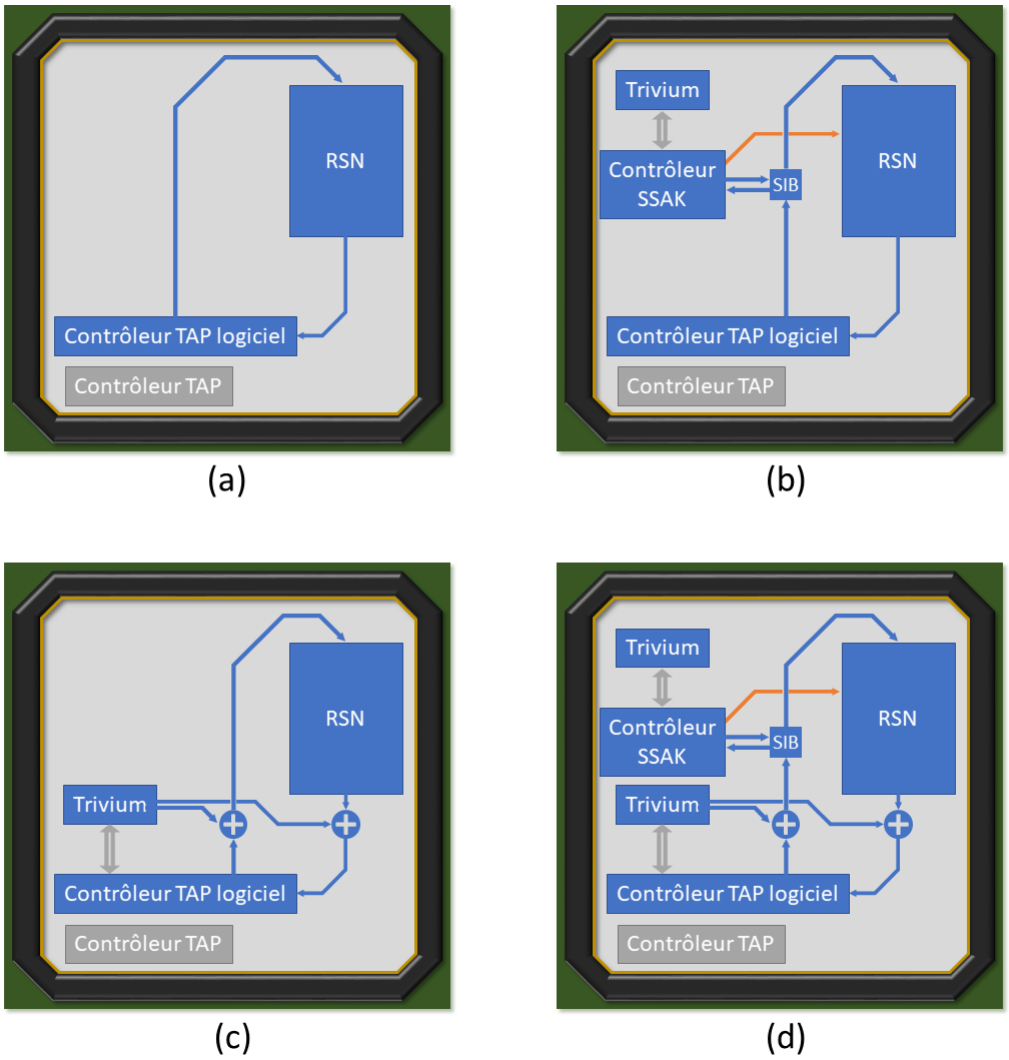


Figure IV.4 Version du démonstrateur SSAK et chiffrement juxtaposés (a), seulement SSAK (b), seulement le chiffrement (c) et version sans protection (d)

La synthèse et l'implémentation de ces différentes versions du démonstrateur permettent d'obtenir des données qui rendent compte de l'utilisation des *Look Up Table* (LUT) et registres *Flip Flop* (FF) dans un FPGA. Pour que les comparaisons aient un sens tous ces démonstrateurs utilisent le même type de cryptoprocesseur, dans ce cas Trivium. Les résultats de ces expériences sont reportés sur le Tableau IV.1.

Tableau IV.1 Ressources FPGA utilisées lorsque les systèmes sont fusionnés, lorsqu'ils sont juxtaposés et le circuit sans protection

	LUT	FF
NON PROTEGE	318	580
SSAK UNIQUEMENT	958	1735
CHIFFREMENT UNIQUEMENT	406	803
JUXTAPOSES	1045	1959
FUSIONNES	958	1736

Les différents résultats présentés sur ce tableau permettent de déduire la répartition des coûts en ressources. La version non protégée permet de connaître la part jointe du RSN et du contrôleur TAP, dénommée RSN par la suite, dans chacune des autres versions. Grâce à cela il est possible de savoir quel est le coût isolé des schémas de protection :

$$LUT_{Contrôleur\ SSAK} = LUT_{Démonstrateur\ SSAK} - LUT_{RSN} = 640$$
$$FF_{Contrôleur\ SSAK} = FF_{Démonstrateur\ SSAK} - FF_{RSN} = 1\,155$$
$$LUT_{Chiffrement} = LUT_{DémonstrateurChiffrement} - LUT_{RSN} = 88$$
$$FF_{Chiffrement} = FF_{Démonstrateur\ Chiffrement} - FF_{RSN} = 223$$
$$LUT_{SSAK+Chiffreur} = LUT_{Démonstrateur\ Juxtaposé} - LUT_{RSN} = 727$$
$$FF_{SSAK+Chiffreur} = FF_{Démonstrateur\ Juxtaposé} - FF_{RSN} = 1\,379$$
$$LUT_{SSAK\ Chiffreur} = LUT_{Démonstrateur\ Fusion} - LUT_{RSN} = 640$$
$$FF_{SSAK\ Chiffreur} = FF_{Démonstrateur\ Fusion} - FF_{RSN} = 1\,156$$

Si la différence de prix entre la solution juxtaposée et la solution fusionnée dans le Tableau IV.1 paressent faible, lorsque les coûts sont répartis entre les différentes composantes comme c'est le cas sur la Figure IV.5, il peut être remarqué que l'économie de ressources représente la totalité du prix du chiffreur de chaîne de scan pris individuellement. Il est à noter que $LUT_{SSAK+Chiffreur} < LUT_{SSAK} + LUT_{Chiffreur}$ et que $FF_{SSAK+Chiffreur} < FF_{SSAK} + FF_{Chiffreur}$; cela peut être expliqué par des optimisations effectué par l'outil de synthèse ou des coûts supplémentaires nécessaires pour effectuer la juxtaposition des deux systèmes. Sur la Figure IV.5 les catégories juxtaposées affichent une répartition estimée entre le contrôleur SSAK et le chiffrement.

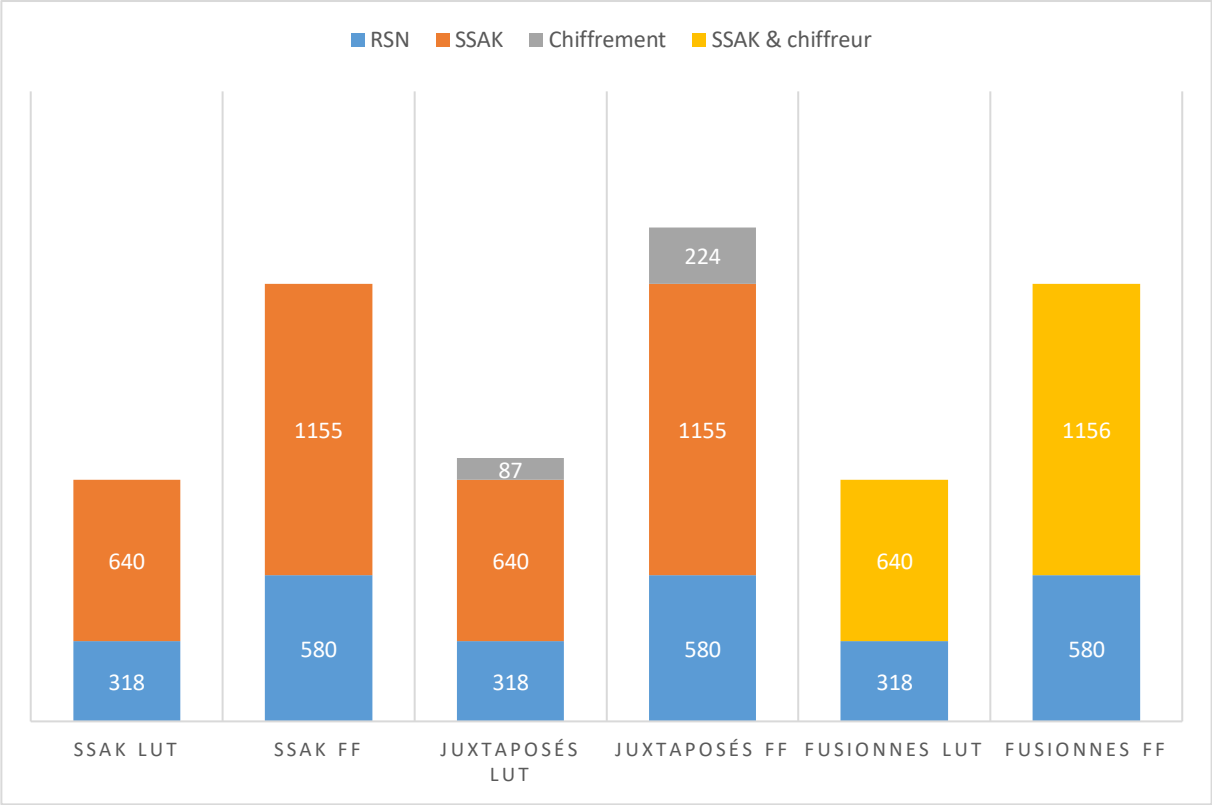


Figure IV.5 Répartition de l'utilisation des LUT dans les solutions juxtaposées et fusionnée

Les travaux effectués pour réaliser ce démonstrateur ont permis de montrer qu'il est possible d'utiliser simultanément les solutions de contrôle d'accès et de chiffrement des chaînes de scan. En outre, il est possible de fusionner certains éléments des deux solutions pour réduire l'utilisation de ressources FPGA. En effet procédant à la fusion décrite dans la section II.6 il est possible d'ajouter au contrôleur SSAK la fonction de chiffrement sans surcoût.

IV.1.3. Authentification et MAST

Cette section décrit le travail fourni pour prouver la faisabilité et la pertinence de l'exécution du protocole d'authentification SSAK par le logiciel MAST pour piloter une infrastructure de test dont une partie est protégée par le contrôleur SSAK. La Figure IV.6 représente le démonstrateur créé mettant en œuvre à la fois les solutions MAST et SSAK. L'une des particularités de ce démonstrateur est que la partie matérielle est en réalité une simulation au niveaux transfert de registre : *Register Transfer Level* (RTL).

Une partie de cette simulation concerne la partie matérielle décrite en RTL et utilise un simulateur de descriptions HDL (ici, ModelSim), la seconde concerne l'exécution de MAST. L'ensemble peut être vu comme un environnement de cosimulation incluant le circuit à tester. L'environnement de test, réalisé par le chercheur à l'origine de MAST, M. Portolan [6], contient le système MAST, un contrôleur TAP et un mécanisme d'échange de données entre les deux simulations.

Les données, c'est-à-dire les vecteurs de test, sont échangés au moyen de fichiers d'échange : MAST écrit un vecteur dans le fichier d'entrée, qui sera lu par ModelSim grâce à un composant VHDL (*Very High Speed Integrated Circuit Hardware Description Langage*) qui pilotera les signaux de commande JTAG du contrôleur TAP. Les vecteurs de test sortant du contrôleur TAP sont eux aussi sauvegardés dans des fichiers avant d'être lues par MAST.

Le circuit testé est issu du travail réalisé pour cette thèse. Il s'agit, comme illustré sur la Figure IV.6, d'un RSN très simple doté, sous un premier SIB, d'un contrôleur SSAK et de son cryptoprocesseur (ici un AES) et sous un S²IB un segment protégé. Ce circuit de test très simple permet de montrer deux choses : il est possible d'utiliser SSAK avec un logiciel de test tiers et le contrôleur développé est compatible avec le standard IEEE 1687. Cela montre aussi la capacité de MAST à s'adapter facilement à des fonctionnalités de test personnalisées et à utiliser des primitives cryptographiques différentes.

Une autre version un peu différente de ce démonstrateur a été présentée dans la publication [46].

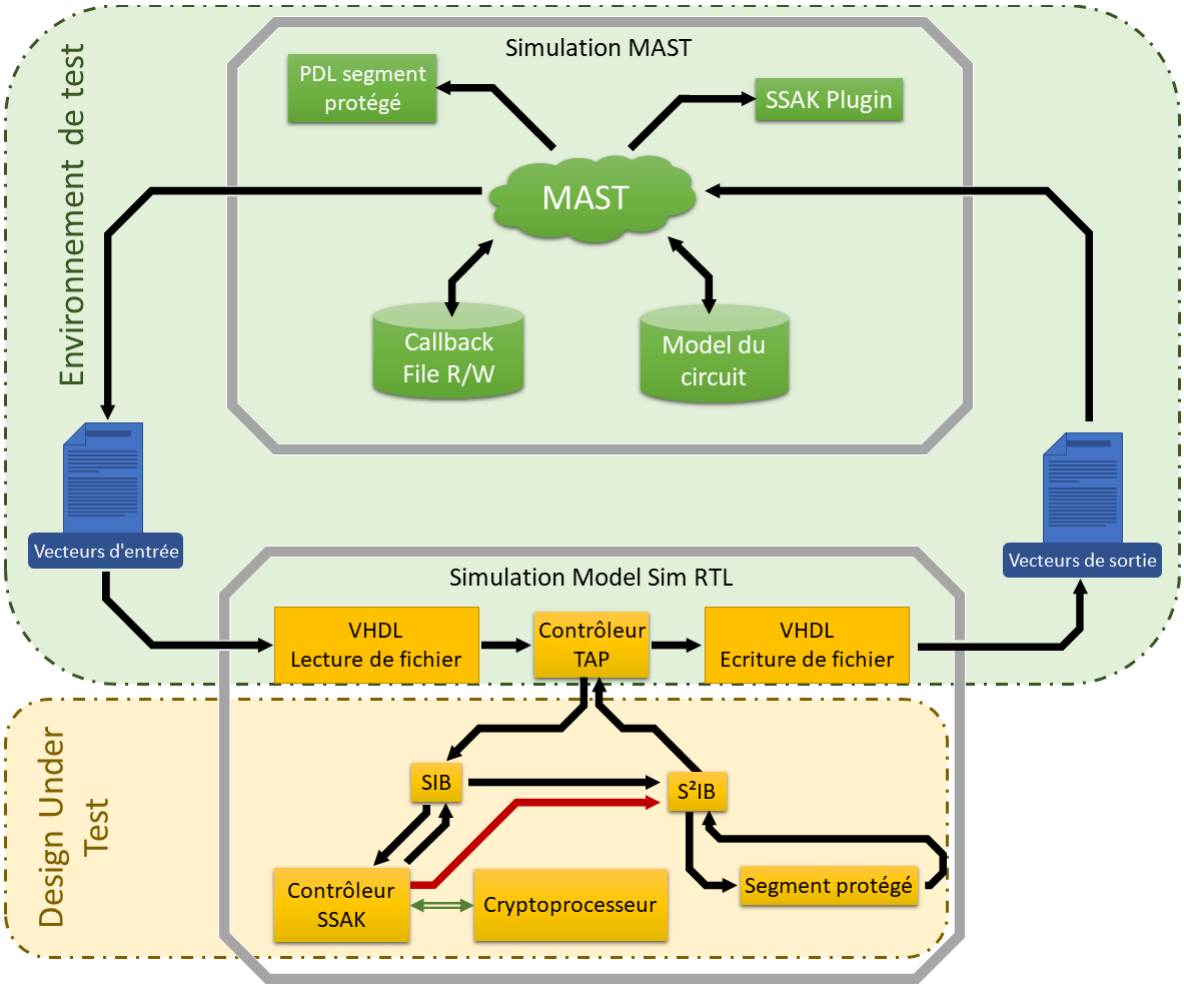


Figure IV.6 Démonstrateur MAST et SSAK

IV.1.4. Chiffrement interne

La section III.2 présente une méthode pour assurer la confidentialité des données entre le testeur et leur segment de destination ou d'origine, en déployant une méthode de chiffrement distribuée dans la chaîne de scan reconfigurable à l'aide notamment de l'introduction des eSIB (*encryption SIB*). Un des problèmes soulevés est la difficulté de gérer les déphasages de clé à appliquer côté utilisateur. S'il est théoriquement possible de gérer ces déphasages grâce à des environnements de test avancés, comme MAST, il semble toutefois pertinent de monter une expérience pour prouver la faisabilité de réaliser ces chiffrements correctement avec une gestion en temps réel.

La Figure IV.7 montre le schéma de fonctionnement de cette expérience. A l'instar du premier démonstrateur, l'expérience est menée sur le FPGA Zynq 7000. Le circuit à tester est un RSN possédant le système SSAK et la technique de chiffrement par eSIB.

Comme expliqué dans la partie III.2.3, les déphasages du flux de clé sont dépendants de l'état courant du RSN, ce qui implique, pour l'utilisateur, d'avoir en permanence la connaissance de l'état de la chaîne de scan. Etant donné qu'il est difficile de développer un modèle représentant l'état du système en comptant seulement sur les vecteurs envoyés, le programme embarqué devant gérer ce modèle a reçu la possibilité de pouvoir directement sonder l'état de chaque SIB directement. Pour cela les SIB ont été adaptés pour qu'ils puissent émettre un signal reflétant leur état, ouvert ou fermé. Ces signaux sont collectés par un périphérique et ainsi disponibles dans l'espace d'adressage du processeur intégré.

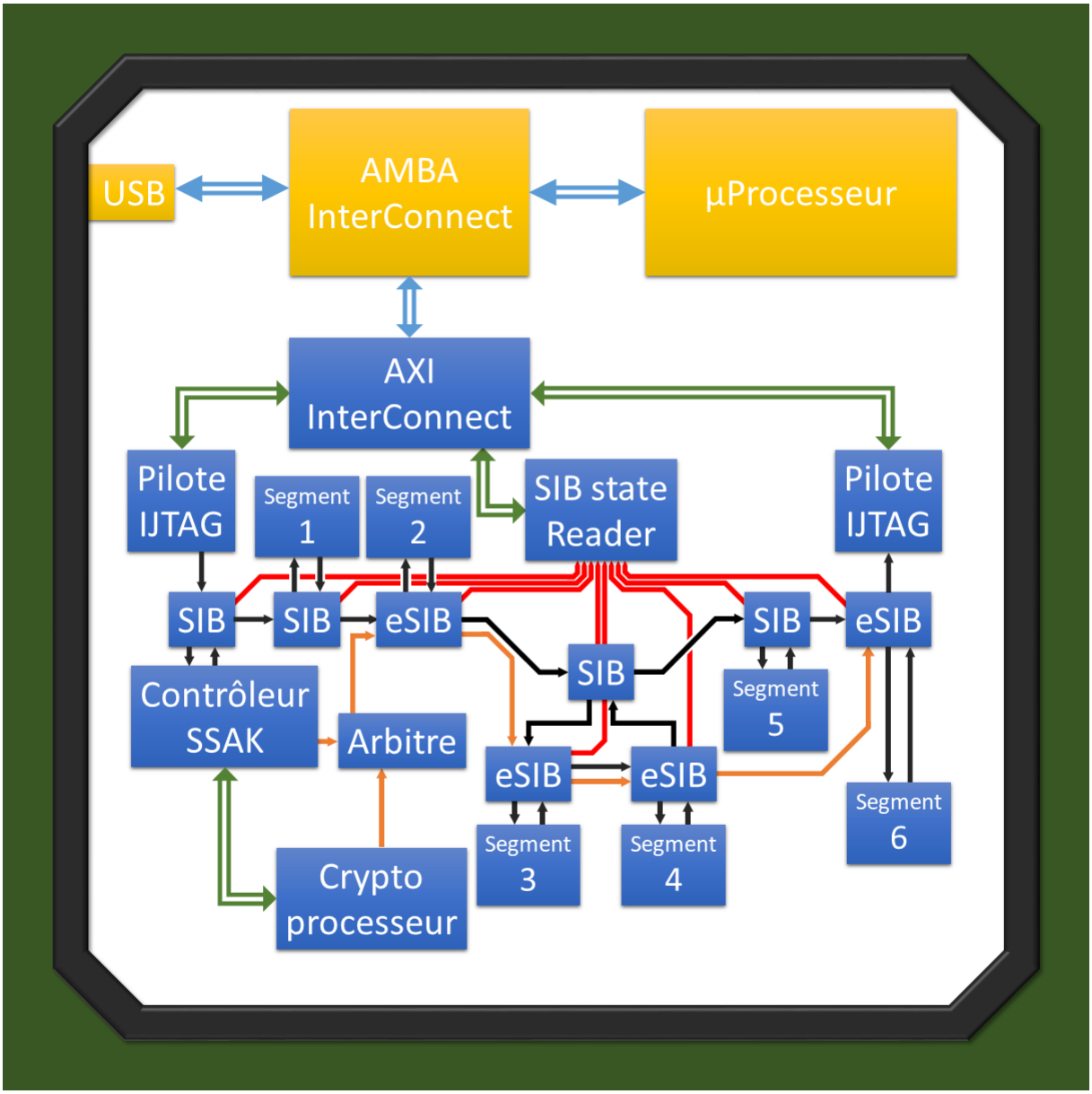


Figure IV.7 Démonstrateur de chiffrement interne

Ce processeur héberge une application de gestion du contrôleur SSAK et du chiffrement interne, qui possède notamment un modèle du réseau de test. Ce modèle est présent sur le diagramme UML de la Figure IV.8. La classe SSAK est la classe qui gère les communications avec la chaîne de scan, l'authentification et le chiffrement de chaîne de scan. Le RSN est représenté à l'aide de quatre classes :

- *scan_el* est une classe abstraite représentant tout élément sur la chaîne de scan

Elle sert de modèle aux classes *scanReg* et *SIB*, elle ne possède qu'un seul membre *globalPos* qui représente la position dans la chaîne de scan. Elle possède des méthodes abstraites, notamment *length()* et *maxLength()* qui permettent respectivement d'obtenir la longueur et la longueur maximale de la chaîne si celle-ci est variable. *updatePos()* permet de mettre à jour la position dans la chaîne de scan.

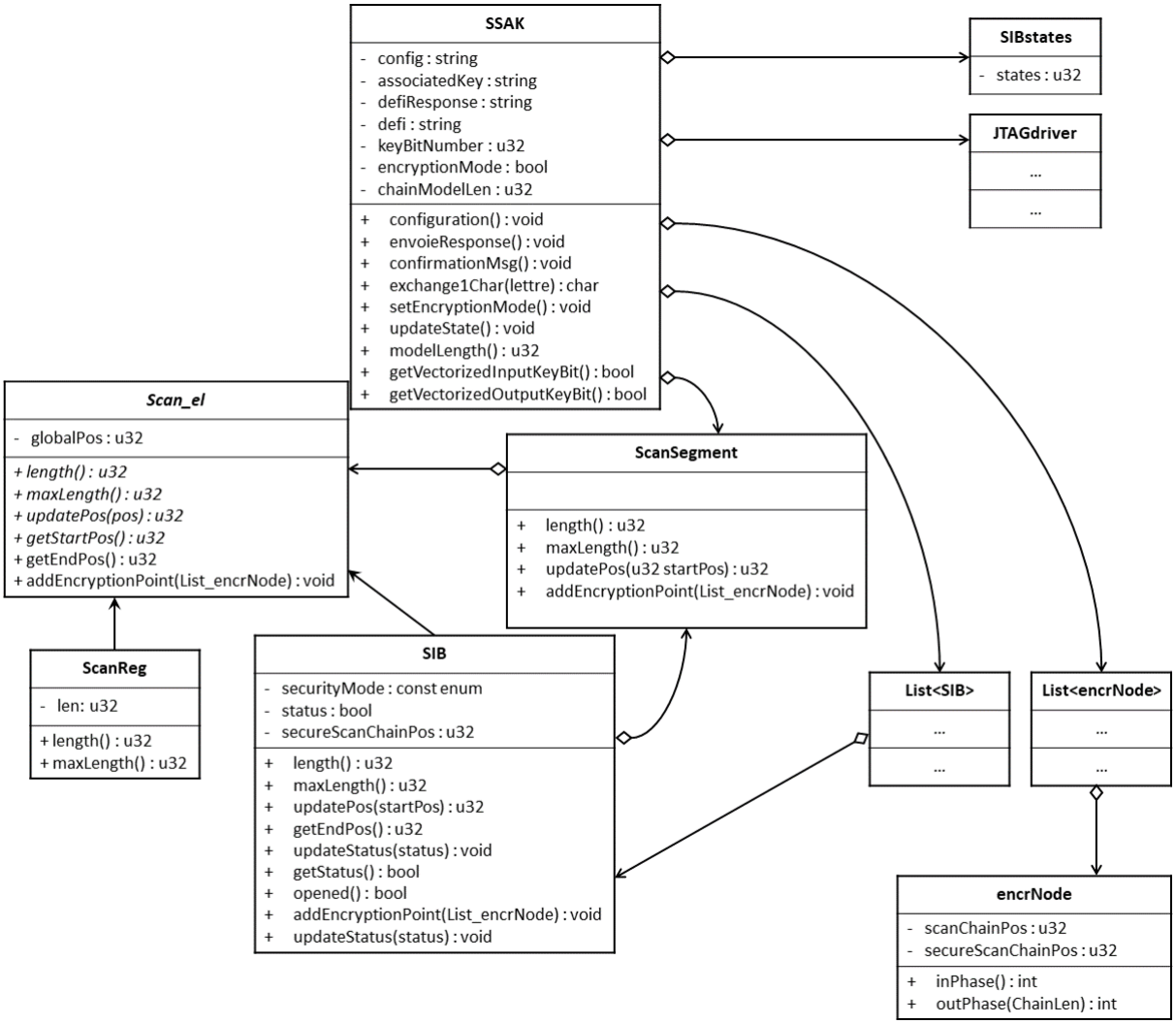


Figure IV.8 Diagramme UML de la gestion de model de chiffrement interne

- *ScanReg* est une classe fille de *Scan_el*.

Elle représente dans la chaîne un registre de scan. Son attribut *len* représente le nombre de bits du registre. Elle redéfinit les méthodes abstraites de sa classe mère ; dans son cas les méthodes *length()* et *maxLength()* retournent toujours la même valeur, la longueur *len* du registre.

- *SIB* est la seconde classe héritant de *Scan_el*

Elle possède plus d'attributs, une constante *securityMode* à trois états permettant de définir s'il s'agit d'un SIB, d'un S²IB ou d'un eSIB, un attribut *status* représentant l'état du SIB, qui est faux lorsque que le SIB est fermé et vrai lorsqu'il est ouvert, et *secureScanChainPos* permettant d'enregistrer la position sur la chaîne de scan sécurisée. Enfin la classe SIB contient un objet de type *ScanSegment*, qui représente le segment de scan hiérarchiquement contenu par les SIB dans les RSN. C'est dans la classe SIB que se distinguent les méthodes *length()* et *maxLength()*. La méthode *maxLength()* retourne la longueur SIB, qui est de 1, plus la longueur maximale du *ScanSegment* contenu, tandis que la méthode

length() retourne la taille du SIB plus la taille du *ScanSegment* seulement si le SIB est ouvert. Les autres méthodes sont détaillées dans les Figure IV.9 et Figure IV.10.

- *ScanSegment* représente comme son nom l'indique un segment de scan.

Cette classe contient uniquement une liste de plusieurs objets de type *Scan_el* positionnés sur le même niveau hiérarchique. Grâce au polymorphisme utilisé, il peut aussi bien s'agir de registres de test que de SIB. Cette classe n'a pas de lien hiérarchique avec celles de la famille *Scan_el*, mais elle possède tout de même trois méthodes nommés *length()*, *maxLength()* et *updatePos(pos)*. Les méthodes *length()* et *maxLength()* appellent pour chaque objet de la liste d'éléments de scan leur fonction homonyme et font la somme des retours.

ScanReg

```
Méthode ScanReg.updatePos(pos)
début
    globalPos = pos
    return pos + 1
fin
```

SIB

```
Méthode SIB.updatePos(pos)
début
    globalPos = pos
    si status = vrai alors
        retourne 1 + pos + segment.updatePos(pos)
    sinon
        retourne pos + 1
fin
```

ScanSegment

```
Méthode ScanSegment.update(pos)
début
    pour chaque élément de la liste d'élements
        pos = pos + élément.updatePos(pos)
    retourne pos
fin
```

Figure IV.9 Pseudo code des méthodes updatePos

Les trois Méthodes dont les algorithmes sont présentés dans la Figure IV.9 permettent de façon récursive de mettre à jour les positions des éléments sur la chaîne de scan ce qui est nécessaire pour être capable de déterminer les phases de clé à appliquer pour effectuer le chiffrement interne. La classe *encrNode* représente un nœud de chiffrement, chaque instance représente un des XOR de chiffrement placé dans les eSIB. Son rôle est de calculer pour son nœud les phases

à appliquer pour les chiffrements des vecteurs à écrire et les déchiffrements des vecteurs lus. Cette classe possède deux attributs :

- *scanChainPos :* la position sur la chaîne de scan normale
- *secureScanChainPos* : la position sur la chaîne de scan sécurisée

Grâce à ces deux données, chaque instance peut calculer un déphasage en entrée et un déphasage en sortie au moyen des méthodes *inPhase()* et *outPhase(chainLen)*. Ces méthodes font appel aux équations de déphasage de clé présentées dans la partie III.2.3. Comme le déphase des clés pour les vecteurs sortants dépend de la position du nœud de chiffrement par rapport à la fin de la chaîne, il faut fournir à la méthode la taille courante de la chaîne.

Pour créer ces nœuds de chiffrement des méthodes basées sur la même technique que *updatePos(pos)* sont employées. Les classes *Scan_el*, SIB et *ScanSegment* définissent ou redéfinissent une méthode *addEncryptionPoint(nodeList)* qui prend pour argument une liste de *encryNode*s. Les algorithmes de ces différentes méthodes sont détaillés dans la Figure IV.10. Les *Scan_el* qui ne sont pas des SIB n'effectuent aucune action. Les SIB, s'ils sont ouverts et sont des eSIB ajoutent deux nœuds de chiffrement, un avant leur sous segment et un autre après. Lorsqu'ils sont ouverts, qu'ils soient eSIB ou non ils appellent la méthode *addEncryptionPoint(nodeList)* de leur *ScanSegment*.

Pour obtenir la liste de nœuds de chiffrement actifs, l'objet principal de type SSAK appelle la méthode *addEncryptionPoint(nodeList)* de son segment en fournissant une liste vide.

Scan_el

$Méthode$ Scan_el.addEncryptionPoint($nodeList$)
$début$
  $retourne\ nodeList$
fin

SIB

$Méthode$ SIB.$addEncryptionPoint(nodeList)$
$début$
  $si\ status = vrai\ alors$
    $si\ securityMode = encryption$
      $node1_{scanChainPos} = getPos()$
      $node1_{secureScanChainPos} = secureScanChainPos$
      $node2_{scanChainPos} = scanChainPos + getLength()$
      $node2_{secureScanChainPos} = secureScanChainPos$
      $ajouter\ node1$ à $nodeList$
      $ajouter\ node2$ à $nodeList$
    $nodeList = segment.addEncryptionPoint(nodeList)$
  $retourne\ nodeList$
fin

ScanSegment

$Méthode\ ScanSegment.addEncryptionPoint(nodeList)$
$début$
  $pour\ chaque$ élément $de\ la\ liste\ d'élements$
    $nodeList =$ élément$.addEncryptionPoint(nodeList)$
  $retourne\ nodeList$
fin

Figure IV.10 Pseudo codes des méthodes addEncryptionPoint

Entre chaque vecteur de test il faut que l'état des SIB soit remis à jour. Pour cela le composant matériel de lecture d'état permet d'obtenir l'état de chacun d'entre eux en lisant ses registres internes. Sur la Figure IV.8 *SIBstate* représente une structure qui pointe sur l'adresse du composant matériel. Grâce à la liste globale de tous les SIB du modèle, la classe SSAK peut mettre à jour l'état de tous les SIB. Ensuite les nœuds de chiffrement peuvent être regénérés.

Ce démonstrateur a permis de prouver qu'il est possible d'effectuer des chiffrements aux bornes des segments desquels on veut protéger les données. De plus ce chiffrement est effectué avec peu de matériel supplémentaire comme cela sera montré dans la partie IV.3.

IV.2. Caractéristiques des cryptoprocesseurs

Comme vu précédemment les cryptoprocesseurs (ou plus largement, les éléments cryptographiques) constituent une des principales demandes de ressources pour le système de sécurité SSAK. Afin d'optimiser au maximum les

coûts il est fondamental d'évaluer différents cryptoprocesseurs et de proposer une palette de compromis entre le coût, la vitesse d'authentification et la sécurité du système. L'objet de cette section est donc de faire une étude d'impact avec plusieurs cryptoprocesseurs.

IV.2.1. Liste d'étude

AES

L'AES est le standard en matière de chiffrement symétrique. Il est basé sur un sous-ensemble de l'algorithme Rijndael [50] pour remplacer l'ancien standard DES qui montrait des signes de faiblesse.

L'AES chiffre les données par bloc, divisés en une matrice de 4x4 mots. Différentes opérations appliquées à cette matrice permettent d'obtenir des résultats intermédiaires. Les tours calculent les résultats intermédiaires et sont constitués de quatre étapes :

- Une substitution bijective de données
- Un décalage de données ligne par ligne
- Une multiplication matricielle par une constante
- Un ajout d'une clef de tours

La première clef (ou les premières pour les versions 192 et 256 bits) sont initialisés par la valeur de la clef principale, puis pour chacune des autres clefs de tours, un calcul à partir de la clef qui la précède permet de déterminer sa valeur. Le standard précise que pour les versions 128, 192 et 256 bits il est nécessaire d'effectuer respectivement 10, 12 et 14 tours [50].

Il s'agit d'un des chiffreurs par bloc les plus sûrs. Même si certaines méthodologies permettent de supprimer 2 bits significatifs [51], c'est sa fiabilité qui lui a permis de devenir le standard et d'être utilisé par le gouvernement américain pour chiffrer les dossiers « TOP SECRET ». Son principal défaut est sa complexité et donc son coût, qui le rend difficile à intégrer dans le contexte des IoT. Mais il est souvent choisi comme référence de comparaison avec les autres cryptoprocesseurs.

Le code source de l'implémentation utilisée pour cette étude vient d'un travail open source [52].

SHA-2

Le SHA-2 est une famille de hacheurs publié en 2002 par la *National Security Agency* (NSA), il existe dans cette famille les hacheurs SHA224, SHA384 et SHA512. Leur fonctionnement est schématisé dans la Figure IV.11

La fonction de hachage SHA-256 travaille avec des blocs d'entrée de 512 bits et calcule une signature de 256 bits. Contrairement au chiffrement, la fonction de

hachage est non réversible ; il n'est pas possible, à partir d'une signature, de retrouver les données fournies en entrée.

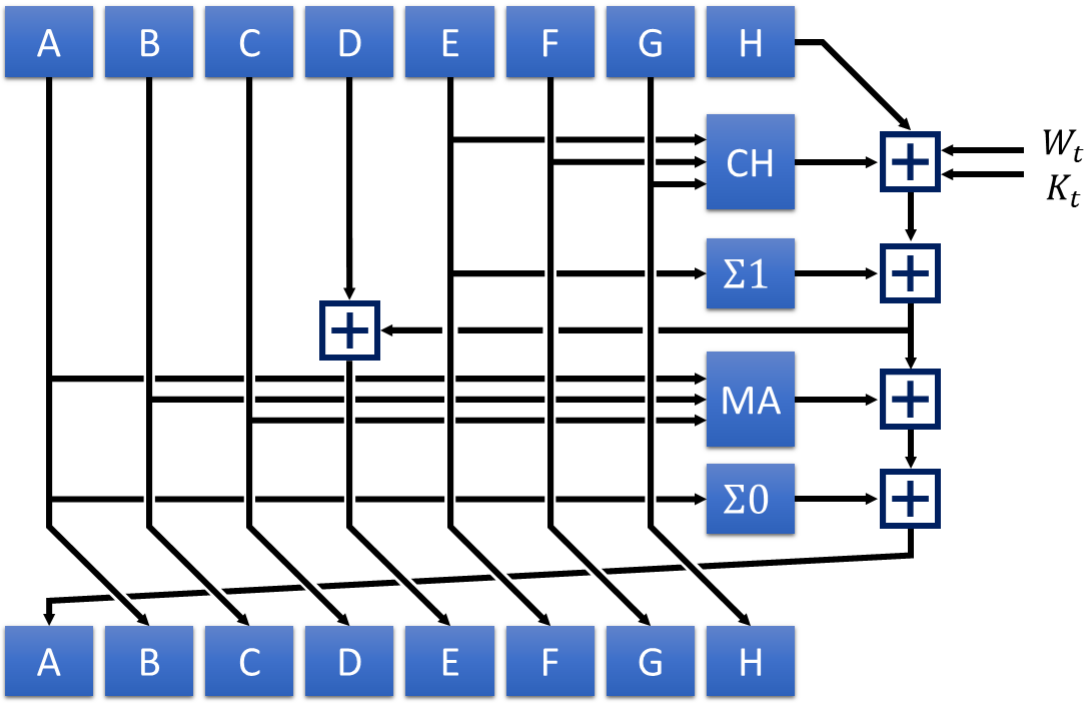


Figure IV.11 Schéma de fonctionnement d'un tour de SHA-2

L'implémentation utilisée pour cette étude a été conçue et testée dans le cadre de cette thèse, il s'agit de la version SHA-256.

SHA-3

En 2012 l'algorithme Keccak remporte la *NIST hash function competition* et est ainsi choisi pour être le SHA-3. Toutefois, ce nouvel algorithme ne remplace pas le SHA-2 qui n'a pour l'instant pas été mis en cause par une attaque. Cependant des attaques efficaces ont été menées sur les hacheurs MD5, SHA-0 et SHA-1 qui ont une architecture voisine de SHA-2. Contrairement à ces hacheurs, le SHA3 est construit sur un modèle très différent appelé construction en éponge comme illustrée par la Figure IV.12. Elle consiste en l'application successive d'une fonction de transformation au bloc d'état.

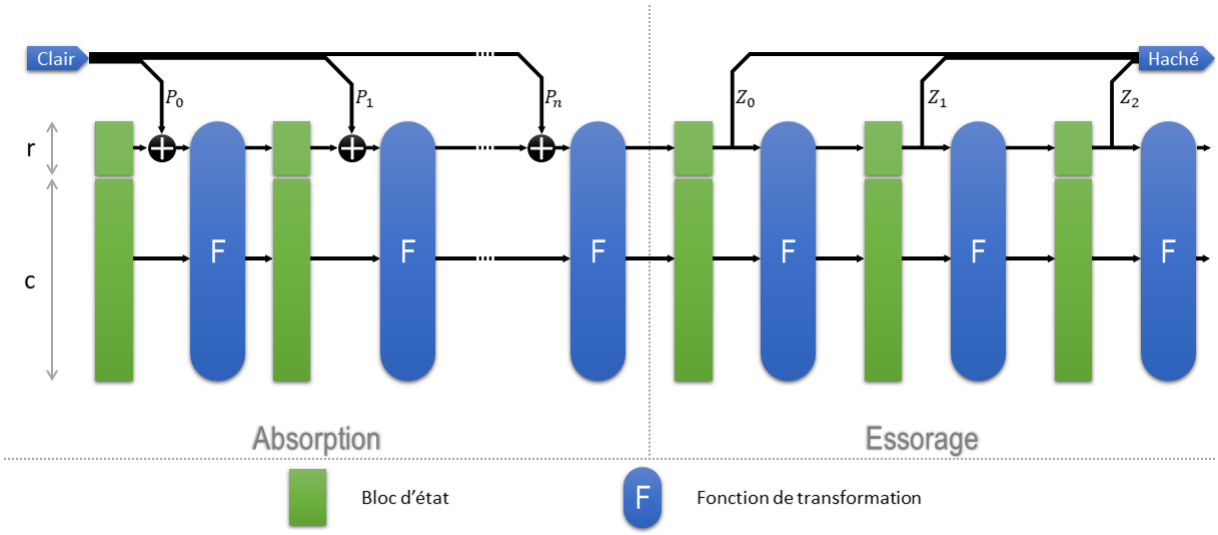


Figure IV.12 Structure en éponge

La fonction en éponge comprend deux phases : l'absorption et l'essorage. Initialement, le bloc de mémoire est initialisé à zéro. Le message d'entrée est divisé et ajusté en blocs de *r* bits ; dans le cas de la version SHA3-256 utilisée ici, ces blocs font 1088 bits soit plus que nécessaire pour absorber l'entrée de 512 bits dans le cadre du protocole SSAK en un seul tour. La phase d'absorption se fait en un ou plusieurs tours de deux étapes. La première étape consiste à effectuer un ou exclusif entre r et le bloc d'entré P_i correspondant, puis le bloc d'état entier subit une fonction de permutation. Ces tours sont répétés jusqu'à l'absorption totale de la clé.

La deuxième phase est l'essorage ; elle se divise aussi en deux étapes. Lors de la première étape un bloc de sortie est prélevé, puis le bloc d'état subit la fonction de transformation et ces étapes sont répétées jusqu'à obtenir une empreinte de la taille désirée.

Lors de l'étude présentée dans cette section deux implémentations différentes de cet hacheur sont utilisées ; l'une vient d'un projet open source [53], l'autre est une version développée dans le laboratoire par K. Jabrane dans le cadre d'un stage financé par l'Institut Cyber@Alpes, ayant l'objectif de détecter des attaques par faute.

TRIVIUM

Le trivium est un chiffreur par flux conçu pour offrir un compris entre coût et vitesse. Une fois configuré avec un vecteur d'initialisation et une clé secrète, tous deux de 80 bits, il peut générer un flot continu de bits pseudo aléatoires Les 2^{64} premiers bits de ce flot peuvent être utilisés comme clé pour chiffrer un flux comme cela est illustré sur la Figure IV.13. L'implémentation du trivium utilisé est issue des travaux de thèse de E. Valea [45].

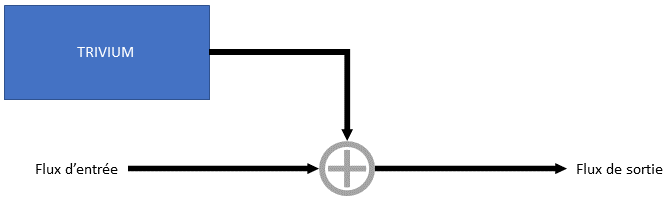


Figure IV.13 Chiffrement de flux avec Trivium

QUARK

Quark est une famille de hacheurs légers. Tout comme le SHA-3, il est basé sur une structure en éponge représentée par la Figure IV.12. La fonction de permutation de cette structure est composée d'un *Linear Feedback Shift Register* (LFSR), de deux *Nonlinear Feedback Shift Register* (NFSR) et de quatre fonctions servant à calculer les rétroactions, comme l'illustre la Figure IV.14. Les NFSR sont chacun initialisé avec la moitié du bloc d'état, le LFSR est quant à lui initialisé par une constante. Afin que la fonction de permutation soit efficace, il faut opérer ces différents registres à décalage pendant plusieurs tours ; la norme spécifie le nombre de tours à cinq fois la taille du bloc d'état.

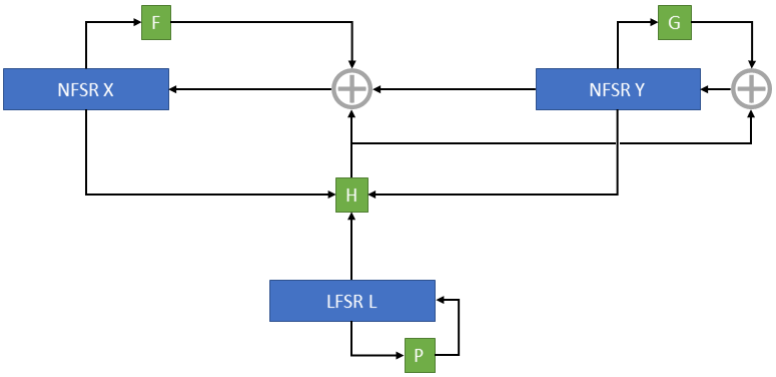


Figure IV.14 Fonction de permutation du hacheur Quark

Par cette définition le QUARK est donc un hacheur qui nécessite beaucoup de tours pour obtenir une signature mais en offrant un faible coût d'implémentation. Le travail de développement et d'implémentation de ce cryptoprocesseur a fait partie du projet de dernière année d'école d'ingénieur Phelma de B. Pollien et A Soussi. En outre ce projet consistait à implémenter un RSN protégé par la solution SSAK en employant ce cryptoprocesseur. Leur développement en VHDL a fourni un QUARK configurable permettant de choisir un compromis entre le coût et la vitesse.

GRAIN

Comme TRIVIUM, GRAIN fut réalisé dans le cadre du projet européen eSTREAM aspirant à favoriser la conception de nouveaux chiffreurs par flux. Un registre à décalage et une fonction de sortie non linéaire sont employés pour générer un flux pseudo aléatoire. La Figure IV.15 réalisée à partir de [54] montre la structure de GRAIN. L'implémentation utilisée pour cette étude a été réalisée par R. Stern, NYU Tandon School of Engineering et disponible sous licence LGPL.

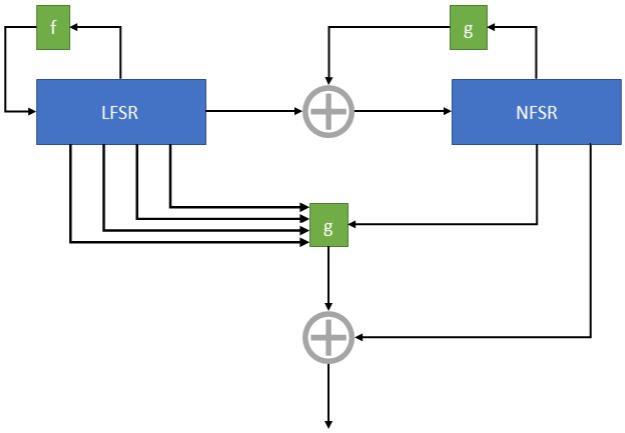


Figure IV.15 Schéma de principe de GRAIN

PRESENT

Le chiffreur léger PRESENT a été développé en 2007, puis adopté en 2012 comme le standard de chiffrement léger (ISO/IEC 29192-2). Il est l'un des moins demandeurs en ressources, en particulier pour ce qui concerne l'implémentation matérielle, car il n'utilise que des fonctions très simples : des « ou exclusifs », des tables de permutation (P-box) et des tables de substitution (S-box) [55]. L'étude présentée par la suite s'appuie sur une implémentation matérielle 80 bits [56].

IV.2.2. Coûts d'intégration

Pour mesurer les coûts d'intégration, une synthèse est effectuée pour chacun des cryptoprocesseurs et de leurs éventuelles variantes pour obtenir une netlist. Celle-ci contient la liste de toutes les portes logiques (combinatoires et séquentielles). Leur nombre permet de rendre compte de l'ordre de grandeur du coût d'un circuit en minorant l'effet de la technologie utilisée lors d'une synthèse destinée à l'implémentation sur puce. La Figure IV.16 résume les résultats obtenus après ces synthèses. Il y a une grande variabilité de ces coûts, jusqu'à un facteur cent ; le choix du cryptoprocesseur est donc primordial afin de maîtriser le coût global d'implantation.

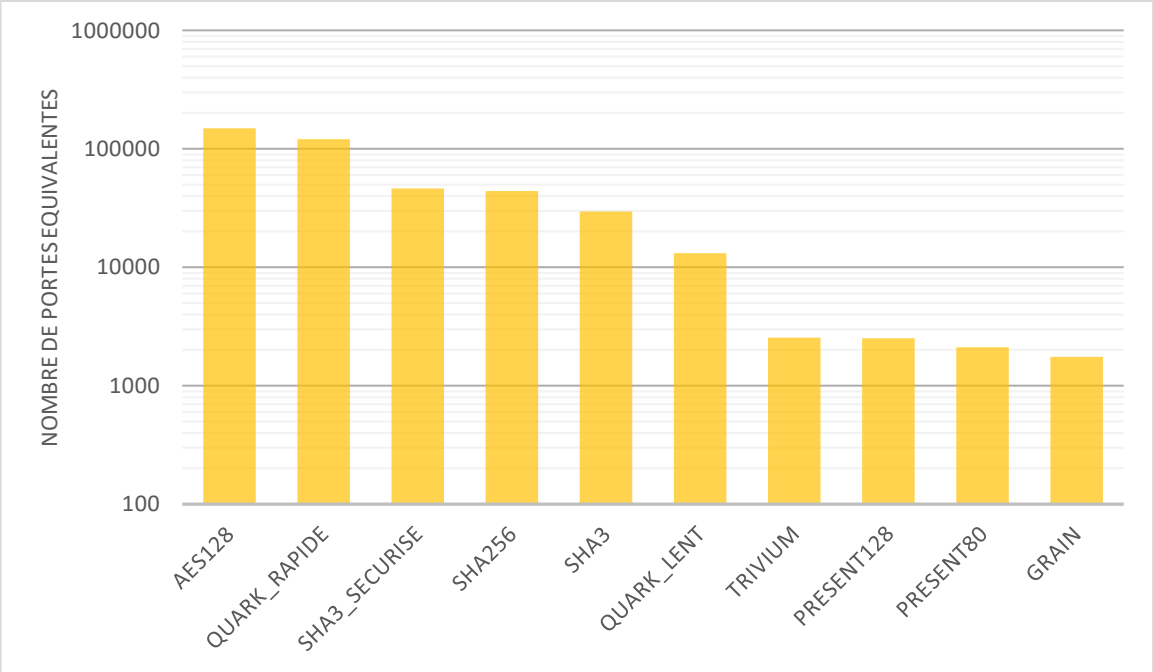


Figure IV.16 Nombre de portes NAND équivalentes utilisées pour chaque cryptoprocesseur

En choisissant les cryptoprocesseurs les moins onéreux par catégorie, la liste serait le PRESENT 80 pour les chiffreurs par bloc, le Quark lent pour les hacheurs et le GRAIN pour les chiffreurs de flux.

Cependant le choix de la solution de chiffrement ne peut pas être pris sur le seul critère des coûts. Il faut également regarder quels sont les performances en termes de vitesse d'opération et de niveaux de sécurité. Ces comparaisons de performances sont proposées dans les parties suivantes.

IV.2.3. Performances

Lorsque le terme de performance est employé concernant les cryptoprocesseurs, cela peut désigner la vitesse de fonctionnement, mais cela peut également se rapporter à leur niveau de sécurité. Dans cette section aucune des deux faces de ce terme n'est mise de côté.

La Figure IV.17 présente les débits binaires des différentes implémentations présentées en partie IV.2.1, obtenues grâce à leur spécification et à des simulations comportementales. Comme pour les coûts, il y a de grandes variations de débit entre elles. L'AES, le plus rapide, a un débit de 12,8 bits par cycle tandis que la version la plus lente du quark ne peut fournir qu'un bit tous les 133 cycles.

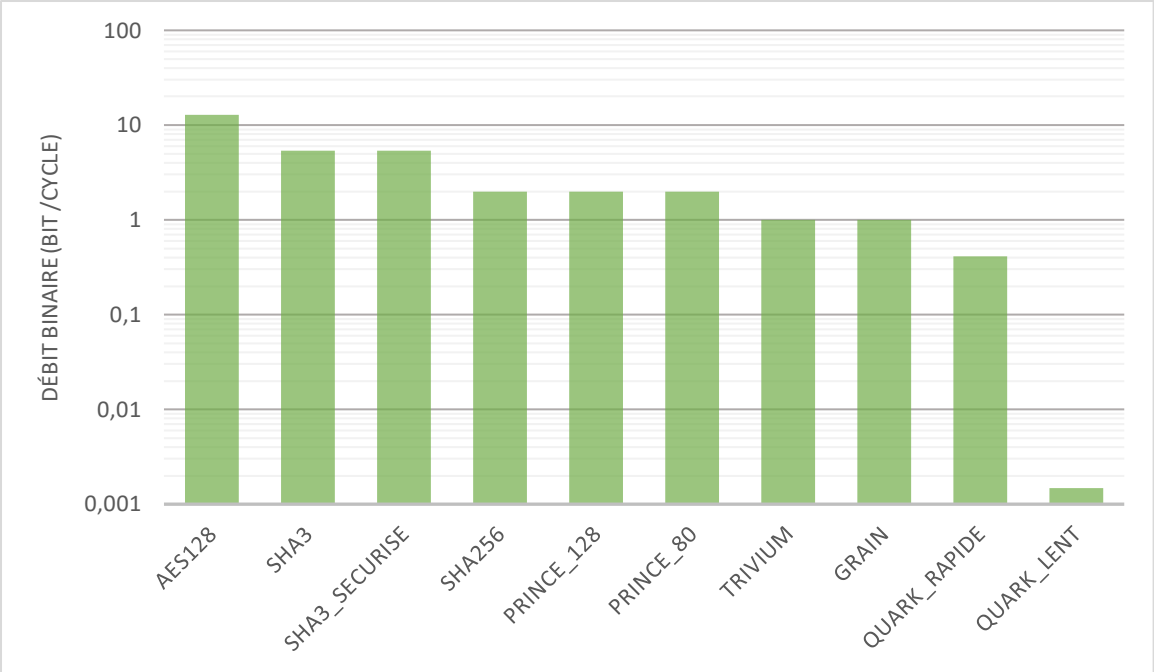


Figure IV.17 Débit binaire des cryptoprocesseurs

Un autre critère à prendre en compte lorsque l'on cherche un cryptoprocesseur est le niveau de sécurité que celui-ci confère. Il n'est pas aisé de mettre particulièrement en avant une métrique de sécurité qui puisse permettre de comparer les chiffreurs par bloc, les hacheurs et les chiffreurs de flux. L'un des choix qui peut être pris est de comparer les tailles de clés. Étant donné que les hacheurs ne disposent pas à proprement parler de clé, elle doit être comprise comme la taille de leur signature. Dans la configuration où SSAK les utilise, cette taille correspond aussi à la moitié du message d'entrée. Le Tableau IV.2 donne ce critère de sécurité ainsi que les autres éléments de comparaison déjà évoqués pour chaque cryptoprocesseur.

Tableau IV.2 Synthèse et performance des cryptoprocesseurs étudiés

Cryptoprocesseur	**Coûts**		**Débit** (bit/cycle)	**Sécurité** (taille de clé)
	Registres	**Portes**		
AES 128	2 965	99 414	12,8	128
SHA 256	3 327	15 374	2	256
SHA 3 Standard	1 643	14 218	5,33	256
SHA 3 Sécurisé	3 238	18 625	5,33	256
QUARK Rapide	1 169	60 937	0,41	80
QUARK Lent	1 201	3 860	0,001 5	80
TRIVIUM	288	659	1	80
GRAIN	194	512	0,125	80
PRESENT 128	263	990	2	128
PRESENT 80	215	843	2	80

IV.2.4. Compromis

Les précédentes parties ont montré les performances et les coûts des cryptoprocesseurs. Il est nécessaire d'avoir une stratégie pour sélectionner de manière judicieuse celui à implémenter. La Figure IV.18 montre de manière graphique les arbitrages qui peuvent être faits. Pour regrouper les données de nombre de registre et de nombre de portes sur un seul axe, la surface d'implantation a été calculée sur la bibliothèque AMS C35. La zone rouge correspond à des débits binaires inférieurs à 1. Il est donc impossible d'utiliser ces cryptoprocesseurs pour effectuer un chiffrement de chaîne de scan. A l'inverse les cryptoprocesseurs dans la zone verte sont inutilement trop rapide pour le protocole SSAK ; en effet le temps d'authentification est minoré par les temps de communication à travers la chaîne de scan. Les meilleurs compromis par catégorie, parmi les éléments analysés, semblent donc être PRESENT pour les chiffreurs par bloc, SHA256 pour les hacheurs et GRAIN pour les chiffreurs de flux.

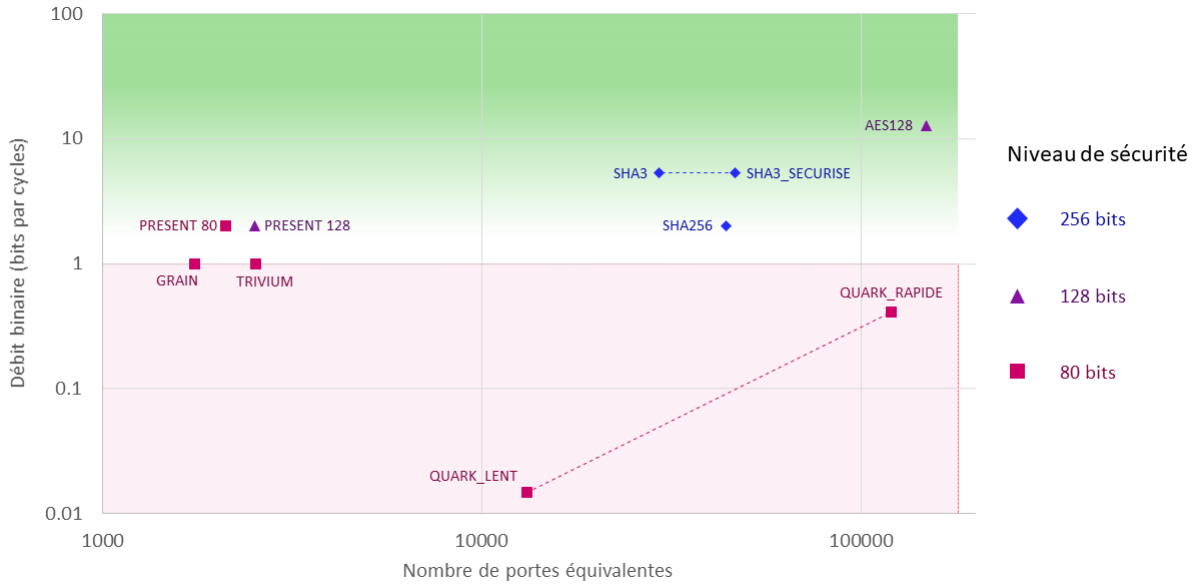


Figure IV.18 Carte d'efficacité vitesse/coût/sécurité des cryptoprocesseurs

IV.3. Etude d'impact

Au-delà des coûts des cryptoprocesseurs, il faut également connaitre le coût total de la solution SSAK, pour pouvoir la comparer aux autres solutions de la littérature. Étant donné qu'une partie des modifications à effectuer dépend de la structure interne du réseau reconfigurable, il faut intégrer les éléments SSAK dans des RSN plus ou moins complexes. La bibliothèque BASTION [57] donne un ensemble de circuits de test pouvant servir à cet usage.

Cette section est divisée en trois parties. La première sert à expliquer quel genre de résultats est souhaité et comment ils permettent de comparer concrètement les différentes solutions. La deuxième partie décrit comment a été mis en place le flot d'évaluation permettant de récupérer les différents éléments de

comparaison. Enfin la dernière partie est consacrée à la présentation de ces résultats et à leur analyse.

IV.3.1. Méthodologie

A l'instar de l'étude sur les cryptoprocesseurs en section IV.2, l'étude d'impact des méthodes de gestion d'accès dans les infrastructures de test se focalisera sur les coûts et les performances des systèmes. Cinq systèmes vont être étudiés : le système basé sur les LSIB et le FGA tous deux présentés dans la partie I.3.4, puis trois variations du SSAK, avec le système de gestion d'accès seul, avec le chiffrement en entrée et en sortie du circuit, et la version avec le chiffrement interne.

Les performances et les coûts des systèmes ne sont pas des constantes, elles dépendent notamment du nombre de segments qui doivent être protégés et de la topologie des circuits. Les résultats gagneront donc en généralité en présentant un balayage du nombre de segments protégés.

IV.3.2. Flot d'évaluation

Pour effectuer les évaluations il est nécessaire de disposer d'un flot d'outils permettant, à partir des fichiers fournis par la bibliothèque BASTION [40], d'obtenir les résultats désirés. La Figure IV.19 représente ce flot, le format de départ est l'« Instrument Connection Language » (ICL) qui décrit comment les instruments de test sont interconnectés. Un parseur ICL a dû être développé ainsi que le simulateur haut niveau car leur rôle est spécifique à cette application.

Deux parties sont nécessaires dans le flot :

- Le flot de synthèse correspond à la partie supérieure de la figure et permet de comparer les coûts d'implantation.
- Le flot de simulation en bas de la figure est utilisé pour avoir les performances en termes de vitesse d'accès aux éléments internes.

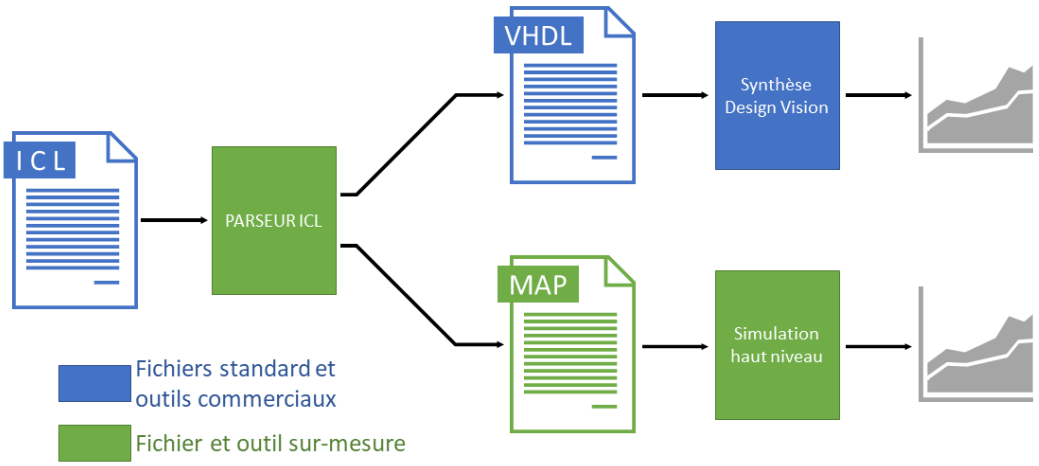


Figure IV.19 Flot de synthèse et de simulation

Le premier outil est le parseur ICL qui, à partir d'un fichier ICL, peut fournir un fichier VHDL paramétrable et un fichier appelé MAP correspondant à une représentation succincte du réseau de scan.

Le fichier VHDL paramétrable permet pour chaque SIB de choisir s'il s'agit d'un S²IB, d'un LSIB, d'un eSIB ou d'un SIB. Dans le cas des LSIB la taille et la valeur des clés peuvent être déterminées individuellement grâce à un fichier de configuration. La Figure IV.20 montre comment sont paramétrés les fichiers de réseau de test. Un des éléments clé est le *configurable SIB* (cSIB). En fonction de la configuration qui lui est donnée il peut se comporter en SIB, en LSIB, en S²IB ou en eSIB. Si le cSIB est dans un mode qui ne nécessite pas de chaîne de scan sécurisée, cette dernière traverse simplement le cSIB. Il prend quatre paramètres :

- *Secure* de type booléen : quand activé, le SIB devient un S²IB.
- *Encryption* de type booléen : quand activé, le SIB se comporte en eSIB.
- *LSIBsize* de type naturel : s'il est supérieur à 1 le SIB peut être considéré comme un LSIB suivi du nombre correspondant de cellules de scan pour accueillir l'emplacement de la clé.
- *LSIBkey* de type vecteur de booléen : il permet de paramétrer la clé d'ouverture du LSIB.

Bien que cela soit rendu possible par les paramètres, plusieurs modes de protection ne doivent pas être activés en même temps. Le fichier RSN rassemble les quatre paramètres de chacun des cSIB dans des vecteurs qui peuvent être remplis dans un fichier de configuration.

Une fois paramétré, ce fichier peut suivre le flot standard d'implémentation en ajoutant, pour les circuits possédant des S²IB, le contrôleur d'authentification. Le logiciel *Design Vision* de Synopsys est utilisé pour faire une synthèse et obtenir les ressources nécessaires.

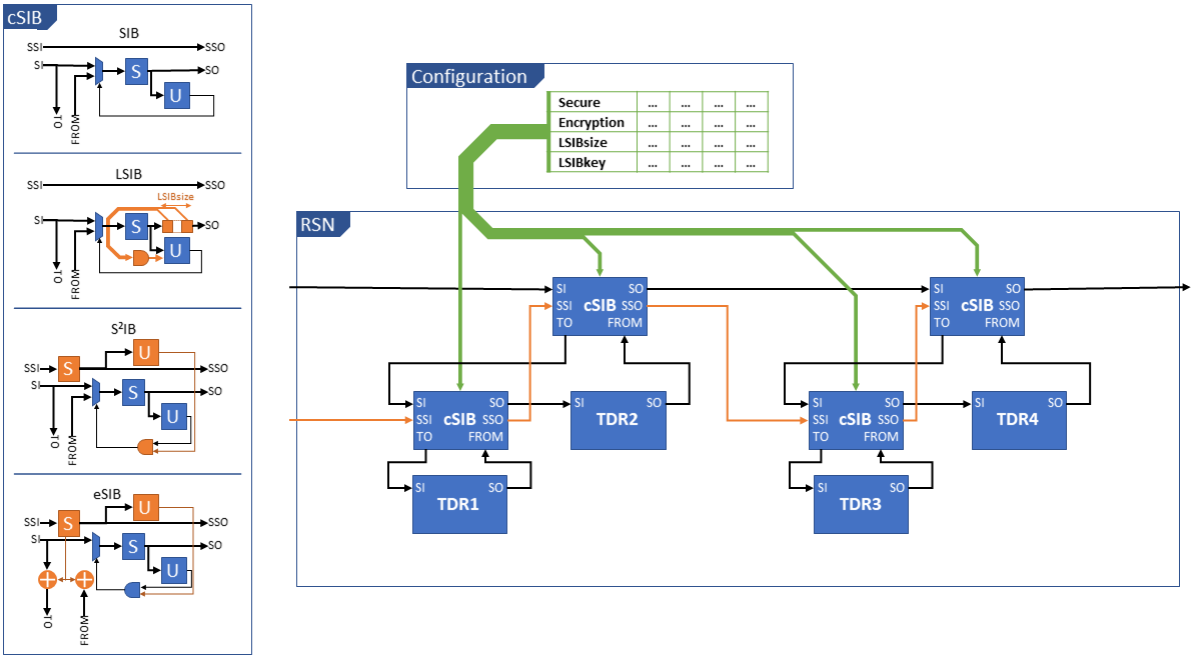


Figure IV.20 Fichier de circuit de test configurable

Le fichier MAP est une représentation du circuit qui n'utilise que deux types d'éléments : les segments et les SIB. Les segments ne possèdent qu'une propriété, leur longueur, et sont définis par elle. Les SIB n'ont pas de propriété mais peuvent contenir d'autres éléments. Un simulateur développé pour l'occasion permet d'évaluer la longueur de la chaîne de scan active en fonction de l'état de ses SIB et ainsi déduire les temps de lecture et d'écriture dans cette chaîne. Il est donc possible d'obtenir, en cumulant les lectures et écritures, le temps d'accès total.

IV.3.3. Résultats

Cette partie présente les résultats obtenus pour les différentes stratégies. Deux caractéristiques sont évoquées : la performance de la solution c'est-à-dire son impact sur le temps de test, et son coût c'est-à-dire la surface supplémentaire qu'elle implique.

Le simulateur de haut niveau temporel permet de mesurer combien de cycles prend un scénario à s'exécuter. Dans le cas de la Figure IV.21 le simulateur mesure le nombre de cycles d'horloge nécessaire à l'ouverture de tous les SIB – protégés ou non – du circuit T512505 [57]. Pour chaque système de sécurité un balayage est effectué pour confronter le temps d'ouverture au nombre de segments protégés. Les résultats montrent que contrairement aux solutions LSIB et FGA, les temps de test ne sont pas affectés par le nombre de segments protégés par SSAK. La solution SSAK est donc très compétitive pour des circuits avec beaucoup de segments à protéger.

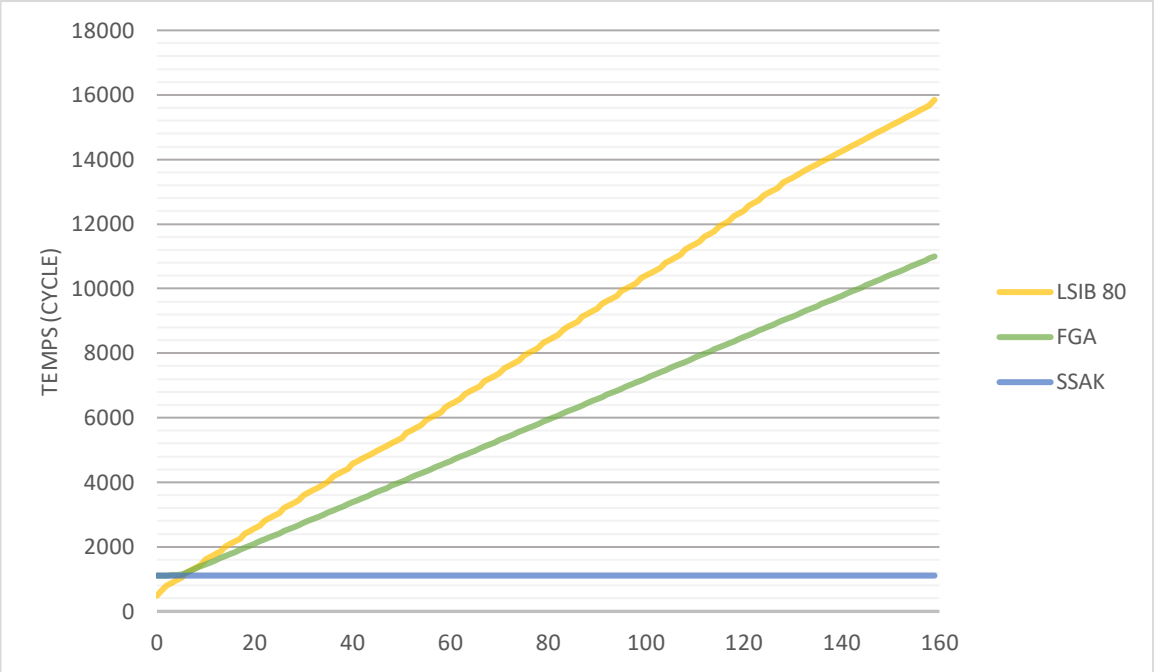


Figure IV.21 Temps d'accès aux segments du circuit T512505 en fonction du nombre de segments protégés avec un hacheur SHA2

La simulation précédente impliquait pour FGA et SSAK un hacheur SHA2, un hacheur rapide par rapport aux autres retenus dans l'étude en section IV.2. C'est l'une des raisons qui rend ces deux solutions si compétitives par rapport à une utilisation des LSIB. Si la simulation est refaite en remplaçant simplement le SHA2 par un quark non optimisé pour la vitesse, les performances des LSIB deviennent plus pertinentes. La Figure IV.22 montre ces résultats, l'axe verticale étant devenu logarithmique pour pouvoir distinguer les courbes LSIB et SSAK. De plus ces résultats montrent que s'il n'est pas envisageable d'utiliser FGA avec un cryptoprocesseur lent, la chose devient possible avec SSAK qui n'a besoin d'effectuer que deux chiffrements.

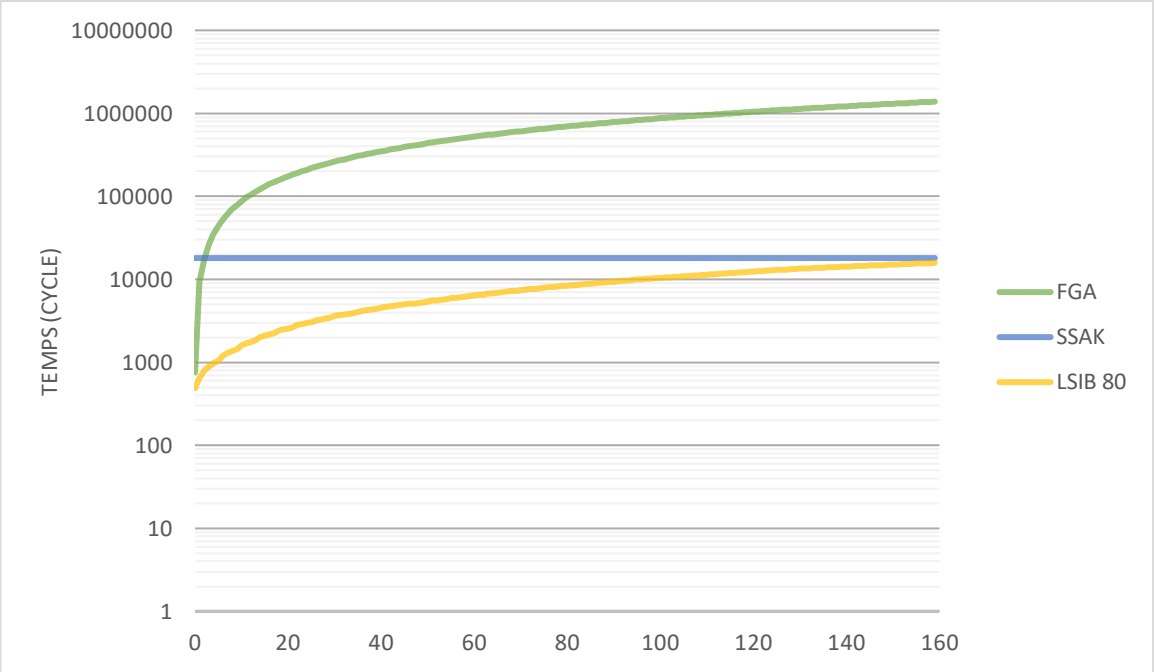


Figure IV.22 Temps d'accès aux segments du circuit T512505 en fonction du nombre de segments protégés en utilisant le QUARK LENT

Grâce aux fichiers VHDL reconfigurables permettant de choisir pour chaque SIB s'il emploie un système de protection et, si c'est le cas, de quel type (LSIB, S²IB, eSIB), il est possible de faire également une étude de coût d'implémentation en fonction du nombre de SIB protégés. La Figure IV.23 montre le coût d'implémentation des différentes solutions de protection en fonction du nombre de segments protégés en utilisant l'AES128 pour FGA, SSAK et eSIB. L'étude s'arrête à 128 bits car les contrôleurs d'authentification voient leur nombre de protections limité à la taille des clés secrètes. Si l'ensemble des protections utilisant ce cryptoprocesseur ont un coût très similaire, elles sont toutes plus cher que le LSIB, bien qu'elles ne dépendent que très faiblement du nombre de protections.

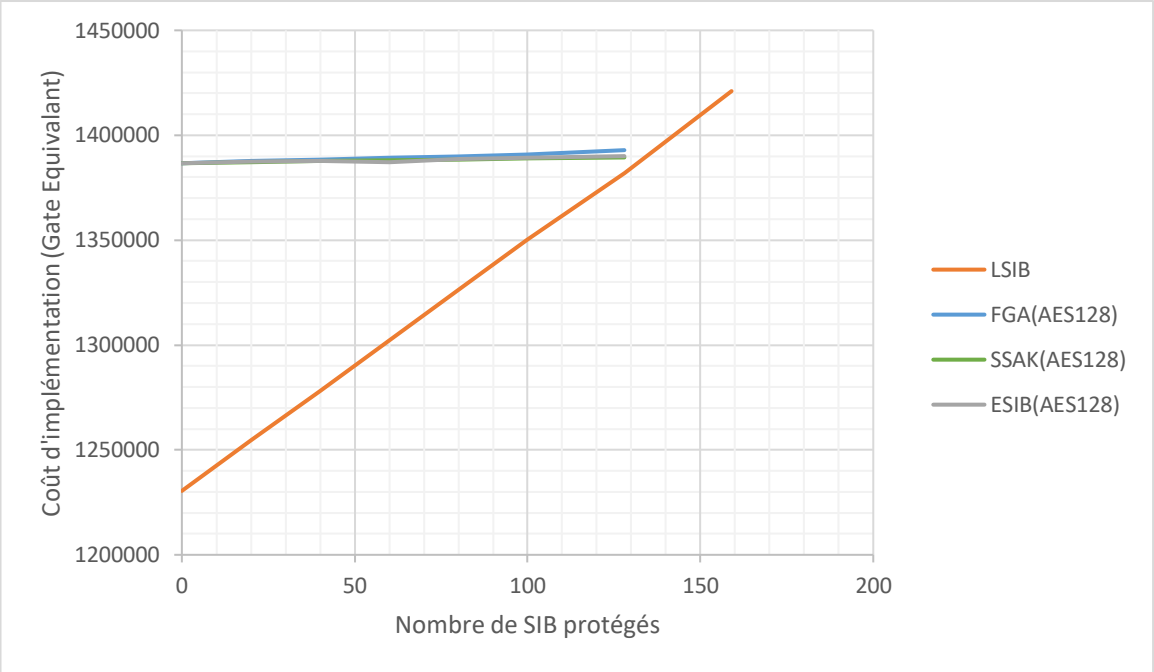


Figure IV.23 Coût d'implémentation des différentes approches en utilisant l'AES128 sur le circuit de stimulation T512505

Afin de rendre plus compétitives ces solutions du point de vue de l'utilisation des ressources, il est possible d'avoir recours à la cryptographie légère. En utilisant un cryptoprocesseur peu cher, par exemple GRAIN, les résultats de FGA, SSAK et ESIB deviennent moins cher, comme montré par la Figure IV.24. Il faut également rappeler que la solution à base de LSIB, du fait de son protocole cryptographique très basique, présente des vulnérabilités majeures et ne propose donc pas le même niveau de sécurité que les autres.

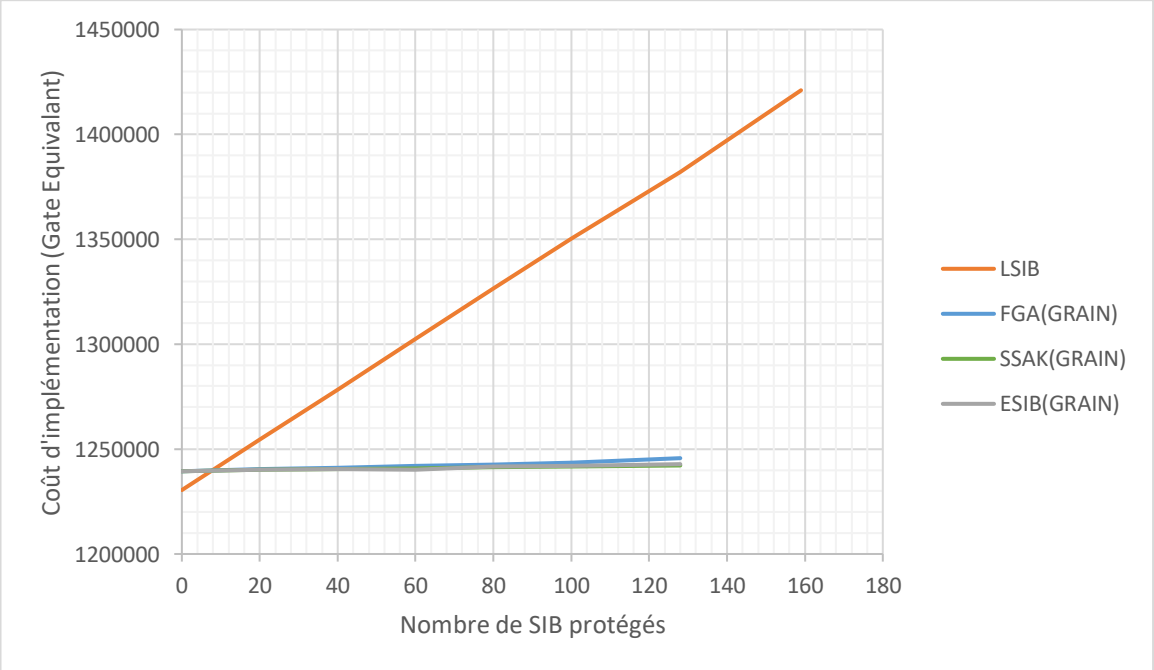


Figure IV.24 Coût d'implémentation des différentes approches en utilisant le GRAIN sur le circuit de stimulation T512505

Afin de mieux comprendre les résultats et voir les différences entre les solutions utilisant des cryptoprocesseurs, la Figure IV.25 résume une étude montrant le coût des différents éléments des solutions, en considérant la protection de cent segments. Il y apparait tout d'abord la contribution du RSN : le coût normal du circuit de test T512505, ensuite la catégorie protection des SIB qui représente le coût supplémentaire nécessaire pour transformer cent des SIB en S²IB, eSIB ou LSIB. La catégorie contrôleur correspond soit au contrôleur SSAK ou FGA excluant le cryptoprocesseur. Les mémoires de clé sont implémentées comme un registre de 128 bits pour les eSIB et SSAK, pour représenter le coût d'une mémoire reconfigurable. Pour la solution FGA il s'agit d'une mémoire ROM 100 x 128 bits générée aléatoirement. La valeur des clés LSIB est quant à elle déjà incluse dans la catégorie protection des SIB. Enfin le dernier élément concerne le cryptoprocesseur, ici un SHA-256.

Dans la situation décrite par cette expérience, les trois solutions à cryptoprocesseurs sont de tailles très proches et faibles comparativement à celle de LSIB. La seule différence notable entre elles est la taille de mémoire de clé, beaucoup plus élevée pour le FGA en raison des 100 clés secrètes qui doivent être conservées.

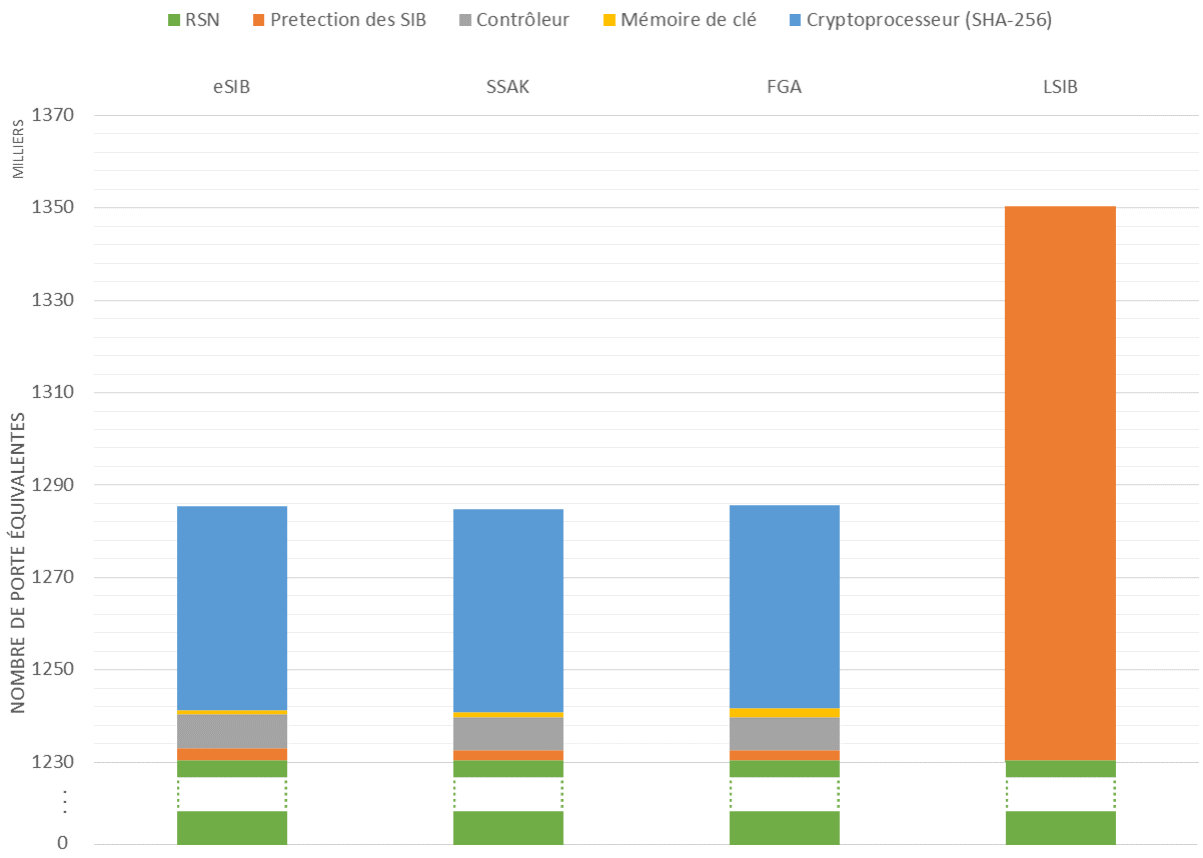


Figure IV.25 Répartition des coûts pour chaque stratégie pour protéger 100 segments dans le circuit t512505

Les résultats apportés dans cette partie démontrent deux choses. Le contrôleur SSAK ainsi que le chiffrement interne ont permis outre de nouvelles fonctionnalités d'améliorer significativement les performances des protections de test, du moins lorsque certains cryptoprocesseurs sont employés. Ces gains en performances sont

permis sans avoir à faire de sacrifice au niveau des coûts d'implémentation qui restent similaires à ceux du FGA voire inférieurs en implémentant le même hacheur. De plus comme montré dans les études de performance, il est possible d'utiliser pour SSAK des cryptoprocesseurs plus lents et moins chers sans que l'impact ne devienne ingérable comme cela serait le cas pour une solution FGA.

Conclusion générale

Les travaux de cette thèse visent à résoudre le problème de sécurité induit par les infrastructures de test. En effet, ces dernières incarnent une faille béante dans tout circuit intégré se voulant sécurisé : sans précaution particulière lors de la conception, un attaquant peut en profiter pour accéder à des données critiques à travers la chaîne de scan. Les travaux se sont d'abord penchés sur l'état de l'art pour connaitre les méthodes utilisées pour défendre les circuits de cette vulnérabilité.

Les travaux réalisés pour protéger des parties ciblées d'un circuit ont conduit au développement de la solution SSAK. Un utilisateur voulant accéder au circuit doit s'authentifier auprès d'un contrôleur dédié grâce à une clé de configuration et des identifiants des parties ciblées. Pour l'authentification, SSAK emploie un protocole défi réponse : l'utilisateur doit utiliser un cryptoprocesseur et sa clé de configuration afin de résoudre ce défi. Le contrôleur doit également résoudre ce défi afin de procéder à la vérification de la réponse utilisateur, mais sans disposer de la clé de configuration spécifique à cette connexion, ce qui serait trop coûteux à stocker. Le contrôleur génère donc cette clé à partir d'une clé spécifique au circuit et de la requête de l'utilisateur. Cette génération procédurale se fait avec l'aide du même cryptoprocesseur que celui utilisé ensuite pour calculer la réponse.

La solution emploie aussi une méthodologie de chiffrement pour s'assurer que les données de test ne soient pas récupérées par un tiers entre l'utilisateur et le circuit. Cette méthodologie issue de la littérature est fusionnée avec le système SSAK pour optimiser les coûts d'intégration et le temps de test. Dans ce but, ils partagent tous deux le même cryptoprocesseur, et utilisent un protocole commun pour procéder à l'authentification d'une part et pour déterminer la clé de chiffrement d'autre part.

Cependant des espions peuvent être placés à l'intérieur du circuit pour récupérer des données de test. Dans ce cas le chiffrement à l'entrée et à la sortie du circuit ne suffisent plus ; il faut que les données sensibles transitent chiffrées également à l'intérieur de la puce pour s'assurer de leur confidentialité. Pour cela des chiffrements sont appliqués à l'entrée et à la sortie de chaque partie protégée du circuit en utilisant la chaîne de scan sécurisée aussi utilisée par le SSAK et des SIB modifiés ("*encryption SIB")*, qui constituent l'une des contributions originales de ce travail.

Pour faciliter l'intégration dans l'industrie, il est important d'adapter la solution SSAK aux flots de test usuels, qui dans l'état actuel rendent très difficiles les tests interactifs nécessaires pour un protocole défi réponse. L'outil MAST, développé dans le laboratoire, permet cette interaction nécessaire. Il est ainsi possible de proposer une solution d'authentification et de chiffrement automatique et transparente, facilitant ainsi son utilisation dans l'industrie.

En plus du développement de la solution et de ses options, une étude a permis de prouver leur faisabilité et d'étudier leur performance et leur coût. Un démonstrateur pour le SSAK a été développé dans chacune de ses versions, afin de faciliter la

diffusion de ces solutions. En parallèle, les différentes versions de SSAK ainsi que des solutions de gestion d'accès issues de la littérature ont été exploitées pour protéger des circuits de test de référence. Des synthèses ont permis d'étudier le coût d'implantation pour chaque solution. Ces études montrent que la solution SSAK a un coût comparable ou inférieur aux solutions de l'état de l'art, et que l'ajout du chiffrement (issu de l'état de l'art, ou celui proposé en interne) ne représente qu'un surcoût négligeable. D'autre part, des simulations comportementales de haut niveau sur les mêmes circuits de référence ont permis d'estimer l'impact des systèmes de sécurité sur les temps de test. L'impact est identique par construction pour les différentes versions de SSAK ; les performances sont équivalentes à celles de l'état de l'art pour un petit de nombre de segments à protéger, mais elles deviennent bien meilleures lorsque le nombre de segments augmente.

Pour continuer à améliorer le système plusieurs pistes sont possibles. La solution proposant un chiffrement interne doit être supportée par MAST et non par le système d'analyse interne proposé dans le dernier démonstrateur présenté. En outre plusieurs éléments du concept SSAK ont été laissés en problèmes ouverts, comme la gestion de clé sécurisée ou la liste noire et sa mise à jour.

Dans les travaux présentés, la gestion de la clé de circuit utilisée par SSAK est laissée à l'unité de gestion des clés du circuit hôte. Il serait également possible de développer un gestionnaire de clé propre à SSAK.

La même problématique existe pour la liste noire d'utilisateurs, bien que les exigences de sécurité ne soient pas les mêmes ; s'il faut empêcher qu'un utilisateur non habilité puisse modifier cette liste, le fait qu'il puisse visualiser cette liste n'est pas forcément un problème. Sa mise à jour peut s'imaginer de différentes manières, par exemple avec un ajout automatique dans la liste noire d'un utilisateur ayant un comportement suspect repéré grâce à une détection d'attaque par chaîne de scan [58], ou par une mise à jour en ligne.

Références

[1] **A. Chandra, K. Chakrabarty**, « A unified approach to reduce SOC test data volume, scan power, and testing time », IEEE Trans. Computer-Aided Design, vol. 20, pp. 355–368, Mar. 2001.

[2] **P. Rosinger, B. M. Al-Hashimi and N. Nicolici**, «Scan architecture with mutually exclusive scan segment activation for shift- and capture-power reduction», IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 7, pp. 1142-1153, 2004

[3] «IEEE standard test access port and boundary-scan architecture», IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001), Mai 2013.

[4] «IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device», IEEE Std 1687-2014, December 2014.

[5] **T. Chen, G. Sunada**, «A Self-Testing and Self-Repairing Structure for Ultra-Large Capacity Memories», Proceedings International Test Conference, 1992

[6] **M. Portolan**, «Automated Test Flow: the Present and the Future», IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019

[7] **F-X. Standaert**, «Introduction to Side-Channel Attacks», Secure Integrated Circuits and Systems, 2009

[8] **P. Kocher, J. Jaffe, B. Jun**, «Differential Power Analysis», CRYPTO 1999. LNCS, vol. 1666, pp. 388-397. Springer, Heidelberg, 1999

[9] **Y. Gao, W. Cheng, H. Zhang, Y. Zhou**, «Cache-Collision Attacks on GPU-Based AES Implementation with Electro-Magnetic Leakages», IEEE International Conference On Trust, Security And Privacy In Computing And Communications / IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2018

[10] **P. Socha, J. Brejník, M. Bartik**, «Attacking AES implementations using correlation power analysis on ZYBO Zynq-7000 SoC board», Mediterranean Conference on Embedded Computing (MECO), 2018

[11] **M. Yoshikawa, Y. Nozaki, K. Asahi**, «Electromagnetic analysis attack for a lightweight block cipher TWINE», 2016 IEEE/ACES International Conference on Wireless Information Technol”Zogy and Systems (ICWITS) and Applied Computational Electromagnetics (ACES), 2016

[12] **S. Skorobogatov**, «How Microprobing Can Attack Encrypted Memory», 2017 Euromicro Conference on Digital System Design (DSD), 2017

[13] **Y. Zhang, X. Zheng,B. Peng**, «A side-channel attack countermeasure based on segmented modular exponent randomizing in RSA cryptosystem», 2008 11th IEEE Singapore International Conference on Communication Systems, 2008

[14] **E. De Mulder, S. Gummalla, M. Hutter**, «INVITED: Protecting RISC-V against Side-Channel Attacks», 56th ACM/IEEE Design Automation Conference (DAC), 2019

[15] **D. Das, S. Maity, S. BinNasir, S. Ghosh, A. Raychowdhury, S. Sen**, «ASNI: Attenuated Signature Noise Injection for Low-Overhead Power Side-Channel Attack Immunity», IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 10, pp. 3300-3311, Oct. 2018

[16] **B. Chevallier-Mames, M. Ciet and M. Joye**, «Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity», IEEE Transactions on Computers, vol. 53, no. 6, pp. 760-768, 2004

[17] **M. Weiner, S. Manich, R. Rodríguez-Montañés**, G. Sigl, «The Low Area Probing Detector as a Countermeasure Against Invasive Attacks», IEEE Trans. Very Large Scale Integr. (VLSI) Syst. vol. 26 no. 2 pp. 392-403, 2017

[18] **Y.-W. Lee, H. Lim, Y. Lee, S. Kang**, «Robust Secure Shield Architecture for Detection and Protection Against Invasive Attacks», IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019

[19] **R. Kapur**, «Security vs. Test Quality Are they mutually exclusive?», International Conferce on Test, 2004

[20] **J. Da Rolt, A. Das, G. Di Natale, M.-L. Flottes, B. Rouzeyre, I. Verbauwhede**, «Test Versus Security: Past and Present», IEEE Trans. on Emerging Topics in Computing vol. 2 no. 1, 2014

[21] **J. Da Rolt, G. Di Natale, M. L. Flottes, B. Rouzeyre**, «New security threats against chips containing scan chain structures», 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2011, pp. 105–110, 2011

[22] **B. Yang, K. Wu, R. Karri**, «Secure Scan: A design-for-test architecture for crypto chips», IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25 no. 10, pp. 2271–2276, 2006
[23] **S. S. Ali, S. M. Saeed, O. Sinanoglu, R. Karri**, «New scan-based attack using only the test mode and an input corruption countermeasure», IFIP Advances in Information and Communication Technology, vol. 461 no. 1, 48–68, 2015
[24] **S. S. Ali, Ozgur Sinanoglu, Samah Mohamed Saeed, Ramesh Karri**, «New Scan Attacks Against State-of-the-art Countermeasures and DFT», 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014
[25] **J. Da Rolt, A. Das, G. Di Natale, M. L. Flottes, B. Rouzeyre, I. Verbauwhede**, « A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures » Proceedings, IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 43–48, 2012
[26] **Y. Liu, K. Wu, R. Karri**, « Scan-based attacks on linear feedback shift register based stream ciphers », ACM Transactions on Design Automation of Electronic Systems, vol. 16 no. 2, 1–15, 2011
[27] **M. Fujishiro, Y. Shi, M. Yanagisawa, N. Togawa**, «Scan-based side-channel attack against symmetric key ciphers using scan signatures», International Conference on Electron Devices and Solid-State Circuits, 2015
[28] **R. S. Chakraborty, S. Narasimhan, S. Bhunia**, «Hardware Trojan: Threats and emerging solutions», 2009 IEEE International High Level Design Validation and Test Workshop, 2009
[29] **M. Rithesh, G. Harish, Bhargav B.V. Ram, Siva Yellampalli**, «Detection and analysis of hardware trojan using scan chain method», 19th International Symposium on VLSI Design and Test, 2015
[30] **K. Hasegawa, M. Oya, M. Yanagisawa, N. Togawa**, «Hardware Trojans classification for gate-level netlists based on machine learning», IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2016
[31] **J. Da Rolt, G. Di Natale, M-L. Flottes, B. Rouzeyre,** « Are advanced DfT structures sufficient for preventing scan-attacks? », IEEE 30th VLSI Test Symposium (VTS), 2012
[32] **J. Da Rolt, G. Di Natale, M-L. Flottes, B. Rouzeyre**, "New security threats against chips containing scan chain structures," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp.110, 2011
[33] **D. Hely, M.-L. Flottes, F. Bancel, B. Rouzeyre, N. Berard, M. Renovell**, «Scan design and secure chip», Proceedings. 10th IEEE International On-Line Testing Symposium, pp. 219-224, 2004
[34] **S. Ahlawat, D. Vaghani, J. Tudu, V. Singh**, «On Securing Scan Design from Scan-Based Side-Channel Attacks», Asian Test Symposium, 2017
[35] **F. Novak, A. Biasizzo**, «Security Extension for IEEE Std 1149.1», Journal of Electronic Testing, vol. 22 pp. 301–303 2006
[36] **J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, M. Thornton**, "Don't Forget to Lock your SIB: Hiding Instruments using P1687," IEEE International Test Conference, 2013.
[37] **A. Zygmontowicz, J. Dworak, A. Crouch, J. Potter**, «Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687 network», Design, Automation and Test in Europe Conference and Exhibition, 2014
[38] **R. Baranowski, M. A. Kochte, H.-J. Wunderlich**, «Securing Access to Reconfigurable Scan Networks», 22nd Asian Test Symposium, 2013
[39] **R. Baranowski, M. A. Kochte, H.-J. Wunderlich**, «Fine-Grained Access Management in Reconfigurable Scan Networks», IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, pp. 934-947, 2015
[40] **E. J. Marinissen, V. Iyengar and C. K. Chakrabarty**, « A set of benchmarks for modular testing of SOCs », International Test Conference, 2002.
[41] «Protecting the FPGA design from common threats», San Jose, CA, USA:Altera, pp. 1-5, 2009
[42] **M. Da Silva, M.-L. Flottes, G. Di Natale, B. Rouzeyre**, «Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption», IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018
[43] **E. Valea, M. Da Silva, M.-L. Flottes, G. Di Natale, B. Rouzeyre**, «Encryption-Based Secure JTAG», IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2019
[44] Nordic Semiconducteur, «KMU - Key management unit», [Online] https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf9160%2Fkmu.html, 2007

[45] **E. Valea**, « Techniques pour la Sécurisation des Infrastructures de Test », thèse de doctorat, LIRMM (Université de Montpelier, CNRS), 2020.
[46] **M. Portolan, V. Reynaud, P. Maistri, R. Leveugle**, « Dynamic Authentication-Based Secure Access to Test Infrastructure », European Test Symposium (ETS), 2020.
[47] **M. Laisne , A. Crouch, M. Portolan,; M. Keim, H.M. Von Staudt , M. Abdalwahab, B. Van Treuren, J. Rearick**,"Modeling Novel Non-JTAG IEEE 1687-Like Architectures", 2020 International Test Conference (ITC20), November 2020, Washington DC, US
[48] **Portolan M., Rearick J., Keim M**., « Linking Chip, Board, and System Test via Standards, European Test Symposium », European Test Symposium (ETS), 2020.
[49] **M. Merandat, V. Reynaud, Emanuele Valea, J. Quevremont, N. Valette, P. Maistri, R. Leuvegle, M-L. Flottes, S. Dupuis, B. Rouzeyre, G. Di Natale**, «A Comprehensive Approach to a Trusted Test Infrastructure», International Verification and Security Workshop (IVSW), 2019
[50] **J. Daemen and V. Rijmen**, "AES Proposal: Rijndael," 2003. [Online]. Available: http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf.
[51] **A. Biryukov, D. Khovratovich and I. Nikolić**, "Distinguisher and Related-Key Attack on the Full AES-256," in Annual International Cryptology Conference, 2009.
[52] **M. Muehlberghuber**, "AES-128", [sourceCode], https://github.com/mbgh/aes128-hdl, 2014
[53] **H. Liao**, "Hardware-implementation-of-Keccak" [sourceCode], https://github.com/HaohaoLiao/Hardware-implementation-of-Keccak, 2018
[54] **M. Hell, T. Johansson**, W. Meier, "Grain - A Stream Cipher for Constrained Environments", eSTREAM submission, 2005, https://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf
[55] **A. BogdanovL. R. KnudsenG. LeanderC. PaarA. PoschmannM. J. B. RobshawY. SeurinC. Vikkelsoe**, «PRESENT: An ultra-lightweight block cipher», 2007
[56] **J. Harttung**, "PRESENT block cipher - VHDL implementation" [sourceCode], https://github.com/huljar/present-vhdl/tree/master/src, 2017
[57] **A. Tšertov, A. Jutman, S. Devadze, M.S. Reorda, E. Larsson, F.G. Zadegan, R. Cantoro, M. Montazeri, R. Krenz-Baath**, "A suite of IEEE 1687 benchmark networks", IEEE International Test Conference (ITC), 2016
[58] **Wei Zhang, Shaohua Teng, Xiufen Fu,** "Scan attack detection based on distributed cooperative model," International Conference on Computer Supported Cooperative Work in Design, 2008

Publications

[a] **V. Reynaud, P. Maistri, R. Leveugle**, « Accès autorisé au réseau reconfigurable de test par ensemble de segments », Colloque du GDR SoC/SiP, 2018
[b] **M. Merandat, V. Reynaud, Emanuele Valea, J. Quevremont, N. Valette, P. Maistri, R. Leuvegle, M-L. Flottes, S. Dupuis, B. Rouzeyre, G. Di Natale**, «A Comprehensive Approach to a Trusted Test Infrastructure», International Verification and Security Workshop (IVSW), 2019
[c] **M. Portolan, V. Reynaud, P. Maistri, R. Leveugle**, « Dynamic Authentication-Based Secure Access to Test Infrastructure », European Test Symposium (ETS), 2020.
[d] **M. Portolan; R. S. Feitoza; G. T. Tchendjou; V. Reynaud; K. S. Kannan; M. Barragán; E. Simeu; P. Maistri; L. Anghel; R. Leveugle; S. Mir**, « A Comprehensive End-to-end Solution for a Secure and Dynamic Mixed-signal 1687 System », 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), 2020

Accès sécurisé aux ressources de test IEEE 1687 dans le contexte IoT.

Résumé

Pour faciliter ou automatiser le test de circuit intégrés de plus en plus complexes, des infrastructures et des instruments de test sont intégrés aux circuits. Cependant, ces infrastructures peuvent aussi être utilisées par des attaquants pour récupérer des informations protégées ou pour induire des comportements non désirés. L'objet de cette thèse est de développer une solution permettant uniquement aux utilisateurs autorisés d'accéder à des blocs critiques du circuit. La solution proposée s'appuie notamment sur la génération procédurale de clés de configuration par segments (nommée SSAK) et un protocole défi-réponse. L'utilisation de clés par segments permet un gain en temps d'authentification par rapport aux méthodes de l'état de l'art, tandis que leur génération procédurale permet de diminuer l'espace de stockage sécurisé dédié aux clés ce qui favorise leur personnalisation et leur mise à jour. En fusionnant avec une solution de chiffrement de l'état de l'art, la confidentialité des vecteurs de test peut être obtenue sans surcoût. Une approche complémentaire s'appuyant sur l'infrastructure d'authentification permet de garantir la confidentialité dans le circuit en dehors du bloc critique testé et ainsi défier des attaques menées à l'aide d'espions internes au circuit. Toutes ces approches nécessitent un contrôle dynamique du test difficilement réalisable avec un flot de test industriel classique. L'utilisation de l'outil MAST, développé au laboratoire, permet de résoudre cette difficulté et d'assurer un test sécurisé fluide et transparent compatible avec les chaînes de production. La faisabilité et le bon fonctionnement des différentes propositions ont été prouvés par un ensemble de démonstrateurs sur support physique (FPGA) ou sur support virtuel (simulations). Ces démonstrateurs ont également permis de comparer les caractéristiques obtenues en termes de surface, temps de test et sécurité avec les résultats de l'état de l'art.

Mots clés : Test sécurisé, Réseau de test reconfigurable, IEEE1687, Authentification, Chiffrement des données de test

Secured access to IEEE 1687 test resources in the IoT context

Summary

To facilitate and automate the testing of increasingly complex integrated circuits, test infrastructures and instruments are integrated inside the circuits. However, those elements are susceptible to be used by malicious users to recover secret data or to alter the normal behaviour of the circuit. This thesis aims at creating a system restricting access to critical parts of the circuit to non-authorized users. The developed solution uses procedural generation of segment configuration keys (called SSAK) and a challenge-response protocol. The usage of configuration keys reduces the authentication time with respect to other methods from the state of the art whereas the procedural generation reduces the requirement of secure memory dedicated to keys, which simplifies their customization and update. By merging with an encryption method from the state of the art, confidentiality of test vectors can be obtained without extra cost. Another developed solution ensures the confidentiality also inside the circuit outside the protected block, and thus counters attacks from integrated spies. All these approaches require dynamic control of the test, hardly achievable with the conventional industrial test flow. The use of MAST, an internally developed tool, allows addressing this issue and ensuring a fluid and transparent secure test compatible with industrial process. The feasibility and the proper functionality of the different proposals have been proven by a set of demonstrators on physical support (FPGA) or on virtual support (simulations). These demonstrators also make possible to compare the characteristics obtained in terms of area, test time and security with state-of-the-art results.

Key words: Secure test, Reconfigurable test network, IEEE1687, Authentication, Test data encryption

Thèse préparée au laboratoire TIMA (Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés), 46 Avenue Félix Viallet, 38031, Grenoble Cedex, France

ISBN: 978-2-11-129268-0