Security Techniques for Test Infrastructures

Emanuele Valea

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En SYAM – Systèmes Automatiques et Micro-électroniques

École doctorale : I2S – Information, Structures et Systèmes

Unité de recherche : LIRMM – Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier

TECHNIQUES POUR LA SECURISATION DES INFRASTRUCTURE DE TEST

SECURITY TECHNIQUES FOR TEST INFRASTRUCTURES

Présentée par Emanuele VALEA

Le 8 juillet 2020

Sous la direction de Giorgio DI NATALE et Bruno ROUZEYRE

Devant le jury composé de

Bruno ROUZEYRE	Professeur à l'Université de Montpellier, LIRMM, Montpellier	Directeur de thèse
Giorgio DI NATALE	Directeur de recherche au CNRS, TIMA, Grenoble	Co-directeur de thèse
Marie-Lise FLOTTES	Chargé de recherche au CNRS, LIRMM, Montpellier	Co-encadrant
Jacques FOURNIER	Ingénieur de recherche, CEA-Leti, Grenoble	Rapporteur
Guy GOGNIAT	Professeur à l'Université de Bretagne Sud, Lab-STICC, Lorient	Rapporteur
Bernd BECKER	Professeur à l'Université de Fribourg, Allemagne	Examinateur
Sophie DUPUIS	Maître de conférences à l'Université de Montpellier, LIRMM, Montpellier	Examinateur
Régis LEVEUGLE	Professeur à l'Université de Grenoble Alpes, TIMA, Grenoble	Président

UNIVERSITÉ DE MONTPELLIER

University of Montpellier

**Graduate School for Information,
Structures and Systems (I2S)**

Security Techniques for Test Infrastructures

by Emanuele Valea

July 2020

Laboratory of Computer Science, Robotics, and
Microelectronics of Montpellier (LIRMM)

Abstract

Test infrastructures are crucial to the modern Integrated Circuits (ICs) industry. The necessity of detecting manufacturing defects and preventing system failures in the field, makes their presence inevitable in every IC and its sub-modules. Unfortunately, test infrastructures also represent a security threat due to the augmented controllability and observability on the IC internals that they typically provide. In this thesis, we present a comprehensive analysis of the existing threats and the respective countermeasures, also providing a classification and a taxonomy of the state-of-the-art. Furthermore, we propose new security solutions, based on lightweight cryptography, for the design of test infrastructures. All proposed countermeasures belong to the category of *scan encryption* solutions and their purpose is to guarantee data confidentiality and user authentication. Each proposed solution is evaluated in terms of implementation costs and security capabilities. The results presented in this thesis, indicate that scan encryption is a promising solution for granting a secure design of test infrastructures.

Résumé

Le test est une étape fondamentale dans le développement des circuits intégrés modernes. Afin de pouvoir tester efficacement des circuits d'une grande complexité interne, il est nécessaire de concevoir des infrastructures de test appropriées au sein de ces circuits. Si ces infrastructures permettent des pratiques de test simples et efficaces et ce, tout au long du cycle de vie du produit, elles offrent également une porte dérobée qui peut être exploitée pour des attaques. C'est pourquoi il est nécessaire de concevoir des infrastructures de test en adoptant une approche axée sur la sécurité. De nombreuses techniques existantes utilisent des implémentations cryptographiques pour empêcher tout accès non autorisé, ou pour assurer la confidentialité et l'intégrité des données de test transmises entre l'utilisateur et le dispositif. Parmi toutes les contre-mesures existantes, l'une des plus prometteuses est le *chiffrement des chaînes de scan* ou *scan encryption*. Cette contre-mesure est basée sur le chiffrement des données de test, ce qui garantit la confidentialité des données et prévient d'une utilisation frauduleuse l'infrastructure de test. Dans cette thèse, nous abordons en détail les techniques de scan encryption. Les contributions de cette thèse peuvent être résumées comme suit:

1. Nous proposons un examen approfondi de l'état de l'art en matière de sécurité des infrastructures de test. Nous proposons une nouvelle classification des menaces de sécurité et des contre-mesures. Sur la base de cette classification, nous effectuons une comparaison entre les contre-mesures existantes, et nous identifions la technique de scan encryption comme une approche prometteuse pour la sécurisation des infrastructures de test.
2. Nous analysons en profondeur les techniques de scan encryption les plus récentes, en identifiant deux approches différentes pour le chiffrement des données de test (c'est-à-dire, respectivement, le chiffrement par flot

et le chiffrement par bloc). En outre, nous mettons en évidence une vulnérabilité affectant les implémentations existantes de scan encryption basées sur le chiffrement par flot.

3. Nous proposons des nouvelles techniques de scan encryption basées sur le chiffrement par flot, en surmontant la vulnérabilité présente dans les propositions les plus récentes. Les implémentations proposées visent différentes infrastructures de test. Leur coût est évalué en termes de surface et de temps de test.
4. Nous comparons les techniques proposées basées sur le chiffrement par flot, avec les techniques existantes basées sur le chiffrement par blocs. Nous soulignons les avantages et les inconvénients des deux implémentations, en fournissant des recommandations permettant aux concepteurs d'opter pour la technique de scan encryption la plus adaptée à leurs besoins.
5. Nous analysons la sécurité des techniques de scan encryption étudiées, et nous identifions une catégorie d'attaques qui peuvent être menées sur les circuits intégrés protégés par la scan encryption. Cette vulnérabilité affecte toutes les implémentations de scan encryption et provient d'un manque de vérification de l'intégrité des données.
6. Nous améliorons la scan encryption, en ajoutant un mécanisme de contrôle d'intégrité léger basé sur des codes de parité. Cette version améliorée de la scan encryption offre une protection complète contre toutes les menaces connues.

1 Test des Systèmes Numériques

Le test numérique est une pratique fondamentale pour détecter la présence de défauts après production des circuits intégrés, il permet d'effectuer un tri entre circuits *bons* et *défectueux*. Les entreprises doivent faire face à des coûts élevés pour garantir un flot de tests efficace, mais cela ne peut être évité. C'est pourquoi les ingénieurs de test font de gros efforts pour garantir une détection maximale des fautes avec un coût de test minimal en termes de développement et d'application sur l'ensemble d'une production. La procédure de test de base consiste à stimuler le dispositif testé avec des séquences de test, qui stimulent la logique interne du circuit intégré

et propagent tout effet de fautes éventuelles vers les sorties. Ainsi, il est possible de détecter la présence d'une faute en comparant les réponses des tests obtenues sur un circuit avec celles dérivées des spécifications. L'objectif est de maximiser la *couverture des fautes* (c'est-à-dire le pourcentage de fautes détectées par rapport au total des fautes possibles), et de réduire au minimum la durée du test. Cependant, la génération des séquences de test et la mesure de la couverture des fautes sont des questions qui sont loin d'être négligeables.

La génération de séquences de test nécessite l'utilisation d'algorithmes spécifiques, appelés *générateurs automatiques de séquences de test* (en anglais Automatic Test Pattern Generator ou ATPG), qui sont capables de traiter la netlist des circuits et de générer un ensemble de vecteurs de test qui sont en mesure de fournir la couverture de fautes la plus élevée. Les algorithmes ATPG fonctionnent très bien et efficacement sur la logique combinatoire, mais il n'en va pas de même dans le cas de la logique séquentielle. C'est ici que les *infrastructures de test* entrent en jeu. Les concepteurs doivent employer la *conception pour la testabilité* (en anglais Design-for-Testability ou DfT) pour pouvoir générer et acheminer les vecteurs efficacement au sein du circuit. Il s'agit de modules supplémentaires qui sont ajoutés durant la conception du circuit intégré.

La technique de DfT la plus courante pour tester les circuits séquentiels est l'insertion des *chaînes de scan*. Dans ce scénario, les bascules du circuit sont remplacées par des bascules dit *bascules scan*. Ces bascules possèdent une entrée supplémentaire, appelée *scan-in*, qui est connectée à la sortie d'une autre bascule du circuit de façon créer un long registre à décalage appelé une *chaîne de scan*. Un signal *scan-enable* permet de faire passer toutes les bascules du *mode fonctionnel* (c'est-à-dire que le circuit fonctionne normalement et que les bascules fonctionnent de manière traditionnelle) au *mode test* (c'est-à-dire que toutes les bascules sont configurées en mode scan, et qu'elles sont déconnectées de la logique combinatoire). La première et la dernière bascule de la chaîne de scan sont connectées aux pins externes du circuit intégré par les pins *scan-in* et *scan-out* respectivement. Cette architecture permet au testeur de décaler directement les valeurs dans les registres internes du circuit en utilisant l'accès scan-in. De même, le pin de scan-out permet au testeur de décaler vers l'extérieur le contenu des registres internes.

Des *standards de test* ont été élaborés par l'*institut des ingénieurs en électricité et en électronique* (IEEE) dans le but de fournir des interfaces et des procédures de test universelles. L'élaboration des normes de test de l'IEEE a commencé dans les années 1980 avec le *groupe d'action conjointe sur le test* (en anglais Joint Test Action Group ou JTAG), qui a commencé à développer la norme IEEE 1149.1, publiée en 1991 [1]. L'objectif du standard JTAG était le développement d'une interface de test pour les circuits intégrés qui peut être accessible lorsque les dispositifs sont soudés sur une carte électronique. Ce travail a abouti à la mise au point de la *porte d'accès au test* (en anglais Test Access Port ou TAP), qui représente aujourd'hui une interface universelle pour les infrastructures de test dans les circuits intégrés. Plus récemment, en 2005, la norme IEEE 1500 a été publiée [2]. Cette norme n'a pas eu la chance d'être associée à un acronyme pratique, c'est pourquoi tout le monde l'appelle encore simplement IEEE 1500. Cette norme a été développée afin de surmonter certaines limitations de la norme JTAG dans les applications liées aux systèmes sur puce (en anglais Systems on Chip ou SoC). Enfin, en 2014, la norme IJTAG (en anglais Internal JTAG) a été publiée [3]. Cette norme introduit un nouveau type d'infrastructure de test qui représente un réel progrès par rapport à la logique du JTAG. L'objectif est de fournir une infrastructure flexible permettant d'atteindre différents types de modules et d'instruments intégrés dans les SoC.

2 Menaces pour la Sécurité des Infrastructures de Test

La DfT contraste de manière décisive avec la nécessité de confidentialité et de restriction d'accès qui est requise dans tout type de circuit intégré. L'infrastructure de test donne généralement à l'utilisateur une contrôlabilité et une observabilité accrues sur les détails internes du circuit. En outre, la configuration en chaîne, typique des infrastructures de test standard, garantit que plusieurs entités indépendantes à l'intérieur du système partagent la même connexion de données. Ce scénario peut entraîner de graves vulnérabilités, lorsque des données sensibles sont déplacées à travers le réseau scan. D'un point de vue général, les menaces impliquant l'infrastructure de test d'un circuit intégré appartien-

nent à deux grandes catégories:

- *Menaces externes*: utilisateur non autorisé qui a le contrôle de l'interface TAP du dispositif.
- *Menaces internes*: dispositif malveillant ou cœur interne implanté dans le système par une entité tierce. Dans ce cas, le dispositif malveillant peut accéder à des données propagées par l'infrastructure à laquelle il est connecté.

Les attaques externes et internes peuvent être classées en différentes catégories, selon le type d'action effectuée par l'attaquant et les ressources auxquelles il a accès.

Dans les menaces externes, tous les composants du système sont censés être fiables. Toutefois, si l'attaquant a un accès physique au dispositif, il est en mesure de se connecter à l'interface TAP et d'exploiter la contrôlabilité et l'observabilité mises en évidence sur le dispositif par l'infrastructure de test. En accédant à l'interface TAP, il est possible d'atteindre potentiellement chaque infrastructure qui y est connectée. Dans la littérature, nous avons identifié trois cibles différentes qui sont exploitées par les attaquants: les chaînes de scan, l'infrastructure de debug, le réseau reconfigurable IJTAG.

La tendance actuelle dans l'industrie des circuits intégrés est la globalisation de la conception et de la production. Pour cette raison, les produits finaux proviennent d'un flot de production qui implique de nombreuses entreprises différentes. Les entreprises de conception sans usine fournissent généralement des cœurs de propriété intellectuelle (IP) aux intégrateurs de SoC. Dans un flot de conception typique, l'intégrateur de SoC assemble tous les cœurs IP, provenant de différents fournisseurs, et conçoit les circuits au niveau du SoC pour garantir une intégration correcte. C'est au cours de cette phase, que l'infrastructure de test au niveau du SoC est insérée. L'infrastructure est connectée aux interfaces de test de chaque cœur IP (par exemple, le contrôleur TAP, l'enveloppe de test IEEE 1500). L'interaction entre toutes ces parties est d'une extrême importance pour la sécurité du matériel. Par exemple, l'intégrateur de SoC ne fait pas nécessairement confiance aux fournisseurs de cœur IP. De même, les vendeurs de cœur IP ne se font pas confiance entre eux. Cependant, les cœurs IP sont généralement connectés à l'infrastructure de test en chaîne. Lorsque le testeur envoie des données à un cœur IP cible par l'intermédiaire de l'interface TAP du SoC, celles-ci sont partagées avec d'autres cœurs IP. Le niveau de

confiance de l'intégrateur du SoC à l'égard des fournisseurs d'IP peut varier selon différents scénarios: 1) les cœurs IP sont vendus à l'intégrateur SoC sans enveloppes de test. Dans ce cas, l'intégrateur SoC doit lui-même envelopper les cœurs IP avec des interfaces de test fiables; 2) les cœurs IP sont vendus à l'intégrateur SoC avec des enveloppes de test. Dans ce cas, l'intégrateur SoC ne peut pas non plus faire confiance à la fonctionnalité correcte des cœurs IP, ce qui fait que les interfaces de test des dispositifs connectés à l'infrastructure de test sont considérées comme non fiables. Les mêmes considérations s'appliquent au niveau de carte électronique, où des circuits intégrés provenant de différents fournisseurs sont montés sur la même carte. En général, lorsque les données de test sont transférées par l'infrastructure de test d'un cœur IP non fiable, il n'est pas certain que celui-ci les traitera selon des règles prédéfinies. Deux menaces possibles ont été envisagées dans la littérature jusqu'à présent:

- le cœur IP non fiable *sniffs* espionne les données qui sont transférées et vole éventuellement des informations sensibles;
- le cœur IP non fiable falsifie les données de test pendant qu'elles sont transférées, et peut éventuellement corrompre leurs informations.

Pour autant que nous sachions, il n'y a aucune trace dans la littérature de dispositifs malveillants qui ont été réellement pris en train de saboter une infrastructure de test. Toutefois, certains auteurs ont publié plusieurs scénarios d'attaque impliquant des dispositifs malveillants exploitant leur connexion à l'infrastructure de test. Ces modèles de menace ont été largement utilisés par les chercheurs afin de motiver leurs contre-mesures.

3 Contre-mesures de Sécurité pour les Infrastructures de Test

Nous nous intéressons dans ce paragraphe aux techniques de sécurité visant les infrastructures de test complexes basées sur les standards de test de l'IEEE. Une classification et une taxonomie de ces contre-mesures est proposée. Nous avons montré que les menaces affectant les infrastructures de test peuvent être divisées en menaces *externes* et *internes*. Dans la

littérature, nous pouvons trouver des propositions de contre-mesures concernant les deux catégories d'attaques: 1) les attaques externes sont principalement contrecarrées par les techniques de *l'authentification de l'utilisateur* et de *la détection des attaques*; 2) les attaques internes sont contrecarrées en accordant la *confidentialité des données*, l'*intégrité des données* et l'*authentification des dispositifs*.

Nous avons divisé ces contre-mesures en sept catégories différentes.

Authentification de l'Utilisateur Cette catégorie de contre-mesures vise à éviter que des entités non autorisées n'accèdent à l'infrastructure de test. Si l'utilisateur n'est pas autorisé, le contrôleur TAP est désactivé et les instructions JTAG ne peuvent pas être exécutées. De cette façon, il n'est pas possible d'accéder aux cœurs IP internes ou au réseau IJTAG reconfigurable. En conséquence, l'exploitation des chaînes de scan internes ou de l'infrastructure de debug est empêchée. Deux catégories de techniques d'authentification ont été identifiées. La première repose sur l'insertion d'un *mot de passe* à l'intérieur du contrôleur TAP afin de le verrouiller ou de le déverrouiller complètement. L'autre catégorie regroupe une série de techniques basées sur des *protocoles de réponse aux défis* mettant en œuvre des primitives cryptographiques.

Authentification des Utilisateurs pour les Réseaux IJTAG Trois catégories de techniques d'authentification pour les réseaux IJTAG ont été identifiées dans la littérature. La première est basée sur des *SIB verrouillés* qui ouvrent l'accès aux régions privées du réseau. La connaissance d'un mot de passe secret est nécessaire pour ouvrir ces SIB. La deuxième catégorie est basée sur les *protocoles de questions/réponse*, ou *challenge/response* en anglais, qui permettent l'accès au réseau (ou à de plus petites parties de celui-ci) uniquement aux utilisateurs autorisés. La dernière catégorie vise à *obfusquer* la structure du réseau, en augmentant la complexité de sa rétro-ingénierie.

Authentification basée sur les Privilèges Les contre-mesures regroupées dans cette catégorie sont une extension des techniques d'authentification des utilisateurs. Dans ce cas, tous les utilisateurs n'ont pas le même type d'authentification, mais ils obtiennent

des privilèges différents sur l'infrastructure de test en fonction du niveau de confiance dont ils disposent.

Confidentialité des Données Lorsque des données sensibles sont échangées entre l'utilisateur et l'appareil, la possibilité d'écoute ou *sniffing* à partir d'une troisième entité malveillante représente une menace. Ce risque est présent à la fois dans un environnement carte et dans un SoC, où l'entité malveillante est un cœur IP interne. En outre, les réseaux IJTAG doivent être protégés lorsque des données confidentielles pourraient être transférées par des instruments embarqués non fiables. Nous avons identifié deux catégories de contre-mesures qui assurent la confidentialité des données d'essai. La première catégorie repose sur le cryptage des séquences de test et peut être appliquée à toutes sortes d'infrastructures. Il s'agit des techniques dites de *scan encryption*. La deuxième catégorie est plus orientée vers la protection des réseaux IJTAG. La configuration du *réseau re-configurable de scan* (en anglais Reconfigurable Scan Network ou RSN) est correctement modifiée afin d'*isoler* les instruments non fiables lorsque des données confidentielles y sont transférées.

Authentification des Dispositifs L'authentification de l'appareil est fondamentale pour lutter contre la présence d'appareils non fiables. L'utilisateur qui communique avec un dispositif cible sur une infrastructure de test doit être sûr que la cible est un dispositif authentique et non un faux dispositif issu d'un processus de contrefaçon. Certaines contre-mesures de ce type ont été proposées dans la littérature.

Intégrité des Données Le fait de garantir l'intégrité de la communication permet à l'utilisateur et/ou à l'appareil de s'assurer que les données échangées n'ont pas été modifiées pendant la transmission. Une technique courante consiste à utiliser un code d'authentification de message (en anglais Message Authentication Code ou MAC) attaché à la fin de chaque message transmis. Le MAC est une signature unique qui est calculée en fonction du contenu du message. L'algorithme MAC le plus utilisé dans ce domaine est le Hash MAC (HMAC). Le HMAC est basé sur des fonctions de hachage, telles que SHA-256. Lorsqu'un message est reçu, l'appareil calcule en interne la signature HMAC du message. Si celle-ci est égale à la signature qui a été reçue en an-

nexe au message, cela signifie que le message est intact. Dans le cas contraire, cela signifie que le message a été altéré. La sécurité de cette primitive réside dans la clé secrète partagée utilisée à la fois par l'utilisateur et par le dispositif pour calculer la signature HMAC.

Détection des Attaques Toutes les contre-mesures présentées jusqu'à présent visent à éviter les attaques sur le système cible. Cependant, il est possible de trouver dans la littérature une catégorie de techniques visant à détecter l'exécution des attaques pendant qu'elles sont en cours. Ceci est réalisé par une surveillance du comportement de l'utilisateur sur la puce. Lorsque le comportement de l'utilisateur est considéré comme illégitime, le système détecte une attaque et passe en mode protection. Les techniques de détection peuvent être divisées en deux catégories. La première catégorie, comprend toutes les méthodes de détection basées sur des règles *statiques*. Dès que ces règles ne sont pas respectées, l'utilisateur est considéré comme un attaquant. La deuxième catégorie comprend les méthodes basées sur l'*apprentissage automatique*.

Les techniques de scan encryption ont connu un développement relativement récent en raison de leurs caractéristiques prometteuses. Du point de vue de la sécurité, elles reposent sur un chiffrement des données basé sur du chiffrement symétrique. Le chiffrement symétrique peut être facilement utilisé pour mettre en place un flot de test sécurisé. En fait, le processus de test peut être considéré comme une communication entre un testeur et un dispositif. Le testeur peut être un utilisateur autorisé accédant à l'infrastructure de test sur le terrain, ou un équipement de test automatique effectuant un test après fabrication. L'appareil cible peut être un circuit intégré ou un cœur IP spécifique à l'intérieur d'un SoC dont l'infrastructure de test interne est protégée. La technique de scan encryption fusionne l'authentification des utilisateurs et la confidentialité des données en une contre-mesure de sécurité unique. En fait, tout dispositif malveillant ou utilisateur malveillant qui tente de sniffer le canal de test n'est pas en mesure de comprendre le contenu du message transmis. En outre, un utilisateur non autorisé qui ne connaît pas la clé secrète qu'il a utilisée n'est pas en mesure de chiffrer avec succès les données de test. Il convient de souligner que le chiffrement du scan d'entrée est toujours présent. Par conséquent, sa fonction de déchiffrement ne peut pas être

évitée le long du canal de test. Pour cette raison, la seule façon de communiquer avec succès avec le dispositif cible est de connaître la clé secrète et de chiffrer correctement toutes les données qui sont introduites par le pin de scan-in.

4 Scan Encryption Sécurisé Basée sur le Chiffrement par Flot

Les techniques de scan encryption existantes sont toutes basées sur le chiffrement par flot TRIVIUM (en raison de sa faible empreinte silicium) et elles représentent apparemment une solution moins coûteuse que les techniques de scan encryption basées sur le chiffrement par blocs. Nous avons montré que les solutions basées sur le chiffrement par blocs doivent être accompagnées de circuits supplémentaires pour être compatible avec l'interface série des infrastructures de test. Ce surcoût est particulièrement évident lorsqu'il s'agit de chaînes de scan multiples. Les solutions basées sur des flots ne présentent pas ce problème, mais les implémentations existantes n'ont pas été développées de manière approfondie comme cela a été fait pour les solutions basées sur des blocs. Pour cette raison, nos recherches se sont orientées vers l'exploration de nouvelles possibilités d'utilisation de chiffrement par flot dans la scan encryption, mais, à notre surprise, certains problèmes ont été découverts. Le chiffrement par flot est considéré comme sûr si deux conditions sont remplies: 1) que le flot de clés ou *keystream* soit imprévisible par l'attaquant; 2) le même flot de données ne doit pas être utilisé plus d'une fois. Si l'une de ces conditions n'est pas respectée, il est possible d'effectuer une attaque, appelée *two times pad*. Les trois implémentations de la scan encryption basée sur les flots, présentes dans la littérature pour sécuriser les chaînes de scan, peuvent toutes être exploitées pour réaliser l'attaque two times pad. Dans la proposition de K. Rosenfeld et R. Karri [4], la clé du chiffrement TRIVIUM est établie par l'utilisateur au moyen d'un protocole de défi-réponse. Le vecteur d'initialization (IV) est fixe. Cela signifie que si l'utilisateur envoie deux fois le même défi à l'appareil, il est sûr que la même clé est utilisée deux fois pour le chiffrement des données de test. Ainsi, l'attaquant peut forcer le dispositif à générer le même flot de clés, même sans connaître sa valeur. La technique décrite dans [5] permet

à l'utilisateur de définir directement la clé de chiffrement du flot. En fait, cette solution ne garantit pas l'authentification de l'utilisateur. Dans ce cas, on ne peut que supposer que l'attaque par scan pourrait être effectuée par un dispositif de sniffage. La vulnérabilité two times pad est présente si l'utilisateur ne modifie pas la clé avant d'envoyer un nouveau message au dispositif. La contre-mesure présentée dans [6] est basée sur un chiffrement par flots dont la clé secrète et le vecteur d'initialization sont soit câblées avec des fusibles, soit produites par une procédure de défi-réponse basée sur des fonctions physiques non clonables (en anglais Physically Unclonable Function ou PUF). Chaque appareil possède un ensemble unique de clés secrètes et de circuits intégrés attribués aux différents instruments du réseau reconfigurable IJTAG. Malheureusement, les auteurs ne précisent aucune précaution prise pour changer les valeurs des clés ou des IV entre les différentes sessions de chiffrement. Par conséquent, l'attaque two times pad peut être effectuée, détruisant l'effet du schéma de scan encryption. Il est clair que les mises en œuvre modernes du scan encryption par flots ont négligé le soin qui doit être apporté à la bonne gestion des clés qu'exige le chiffrement par flots. En revanche, la scan encryption par blocs ne souffre pas d'une telle faiblesse. En fait, le chiffrement par blocs peuvent réutiliser la même clé tout au long de la durée de vie du dispositif sans entraîner de problèmes de sécurité connus. C'est une première indication du fait que préférer la scan encryption basée sur le chiffrement par flots n'est pas aussi évident qu'il n'y paraît à première vue. Cependant, nous avons décidé de donner une autre chance au chiffrement par flot et nous avons proposé une nouvelle façon de mettre en œuvre la scan encryption basée sur le chiffrement par flots.

La contre-mesure proposée consiste à ajouter deux chiffrements de flot à l'entrée et à la sortie de la chaîne de scan respectivement. Un attaquant qui ne connaît pas la clé secrète utilisée pour le chiffrement des données de test n'est pas en mesure d'insérer les séquences de test souhaitées à l'intérieur du circuit. De plus, l'attaquant n'est pas en mesure de comprendre le contenu des données de test qui sont transférées hors de l'infrastructure de test. Seuls les utilisateurs ayant connaissance de la clé secrète sont autorisés à accéder aux chaînes de scan. En réutilisant la gestion de la clé du circuit d'origine, la solution n'introduit pas de nouveaux problèmes dans la manipulation de la clé secrète. Le dispositif fournit

une clé de scan encryption dédiée aux utilisateurs autorisés. L'IV utilisée pour initialiser le chiffrement de la chaîne est générée par un générateur de nombres aléatoires véritables (en anglais True Random Number Generator ou TRNG). Ce IV aléatoire est envoyé à l'utilisateur externe par l'intermédiaire du pin de scan-out. De cette façon, l'utilisateur est en mesure de chiffrer correctement les séquences de test. L'IV est totalement aléatoire et il est différent après chaque réinitialisation du circuit, mais il n'est pas secret. Le seul secret est la clé, qui n'est connue que des utilisateurs autorisés. Comme les chiffreurs par flots sont initialisés avec une IV différente à chaque initialisation du dispositif, le même flot de clés n'est jamais généré deux fois. Ainsi, cette solution n'est pas vulnérable à la faiblesse du two times pad, ce qui empêche tout attaquant d'effectuer des attaques de scan différentiel. La première étape de la nouvelle procédure de test consiste à générer des vecteurs de test pour le circuit testé, et à calculer les réponses de test attendues par simulation. Avant toute opération de scan, les chiffreurs par flots sont initialisés en générant un nombre aléatoire utilisé comme IV. Le testeur récupère l'IV aléatoire généré par le biais du pin scan-out, et il chiffre les vecteurs de test hors puce en utilisant l'IV aléatoire et la clé secrète. Une fois l'initialisation du chiffrement par flots terminée, le testeur peut transférer les vecteurs de test chiffrés dans l'infrastructure de test. Chaque vecteur de test chiffré est d'abord déchiffré à l'aide du flot de clés d'entrée généré par le chiffreur par flot. Ensuite, il est déplacé dans les chaînes de scan de l'appareil. Les vecteurs de test sont appliqués au circuit testé et des réponses de test sont obtenues. Pendant l'opération de décalage, les réponses de test sont chiffrées à l'aide du flot de clés généré par le chiffrement par flots placé à la sortie du scan. Les réponses de test chiffrées sont décalées hors du circuit afin d'être déchiffrées hors puce par le testeur. Une fois déchiffrées, les réponses au test peuvent être comparées avec les réponses attendues.

Nous avons montré que les solutions de scan encryption basées sur le chiffrement par flot sont un choix attrayant pour assurer la sécurité des infrastructures de test. Cela est vrai à condition de gérer correctement la génération des IV, contrairement à ce qui avait été proposé précédemment par d'autres auteurs. Cependant, la génération des IV aléatoires n'est pas gratuite. Dans les techniques présentées, nous avons proposé la mise en œuvre d'un TRNG pour générer un IV toujours différent. Comme les TRNG sont souvent

déjà présents à l'intérieur des circuits intégrés, nous ne considérons pas nécessairement cet élément comme un coût exclusivement dû à la présence de la scan encryption. À ce stade, nos recherches ont été guidées par le désir de donner une image claire aux ingénieurs intéressés par la mise en œuvre d'une contre-mesure de scan encryption. Nous résumons ici les conclusions que nous avons tirées de la comparaison des techniques de scan encryption par flot et en bloc :

Sécurité La scan encryption par flot et en bloc assure la sécurité contre les attaques externes et les attaques internes basées sur le sniffage de données de test. Dans le cas des attaques par scan différentiels, la solution proposée basée sur le flot est la seule solution de scan encryption sécurisée existante de ce type, car elle ne permet pas l'attaque two times pad. Les deux solutions nécessitent une clé secrète intégrée dans le dispositif cible, dont la gestion peut bénéficier de la politique de gestion des clés fournie par le dispositif cible.

Testabilité les deux techniques garantissent la testabilité complète du dispositif cible, tant à la fabrication que sur le terrain. Le test de fabrication n'est pas affecté par la perte de couverture des fautes induite par l'ajout de la solution de scan encryption. La scan encryption par flot a besoin d'une méthode permettant de contourner la génération aléatoire du IV lorsque le test in-wafer est effectué. Le test sur le terrain est accordé, car l'accès à l'infrastructure de test est laissé ouvert, même si seuls les utilisateurs autorisés peuvent communiquer avec succès avec l'infrastructure de test.

Coût en Surface la scan encryption par flot mis en œuvre avec le chiffrement par flot TRIVIUM est égal à la moitié du coût en surface induit par la scan encryption en bloc mise en œuvre avec le chiffrement par bloc SKINNY (c'est-à-dire le chiffrement par bloc le plus léger que nous avons utilisé dans nos expériences). En fait, il est possible d'observer qu'un chiffrement par blocs léger, tel que le chiffrement SKINNY, a la même empreinte de surface que le chiffrement TRIVIUM. Toutefois, dans le cas de scan encryption basée sur des flots, il est suffisant d'implémenter un seul chiffrement générant deux flots de clés, alors que dans le cas de chiffrement par blocs, il est obligatoire d'implémenter un chiffrement lors du scan-in et un chiffrement lors du scan-out. Il en

résulte que la scan encryption par flot coûte la moitié de la scan encryption en blocs. D'autre part, si l'on considère le coût du TRNG comme un coût induit par la mise en œuvre de la scan encryption par flot, son coût augmente considérablement et devient jusqu'à trois fois supérieur à celui d'une solution par blocs. Par conséquent, il est clair que la scan encryption par flot n'est pratique du point de vue du coût de la surface que si un TRNG est déjà disponible dans l'implémentation du dispositif cible.

Coût en Temps de Test la scan encryption par flot a un grand avantage du point de vue du temps. Cela est possible parce que le chiffrement par flot bénéficie de son adéquation naturelle avec l'interface série des infrastructures de test standard. Pour cette raison, la pénalité de temps de test induite par la solution basée sur le flot est un temps d'initialisation constant qui est négligeable par rapport au temps de test total du dispositif cible. Les solutions de scan encryption par blocs doivent adapter le chemin de données des chiffrages par blocs à l'interface série de l'infrastructure de test. Pour cette raison, la nécessité de compléter les vecteurs de test par des bits supplémentaires pénalise fortement le temps de test. Cette surcharge peut être réduite par l'ajout de points de test, modifiant ainsi l'infrastructure de test du dispositif cible.

Chaînes de Scan Multiple lorsque plusieurs chaînes de scan sont présentes, la scan encryption par flot peut bénéficier de sa capacité à générer plusieurs flots de clés à partir du même chiffrement par flot. Pour cette raison, il est possible de gérer jusqu'à 32 chaînes de scan mettant en œuvre un seul chiffrement de flot TRIVIUM. Dans le cas de la scan encryption en bloc, la gestion de plusieurs chaînes de scan est définitivement plus compliquée. En effet, les chiffrages par blocs doivent être pilotés par une fréquence d'horloge plus élevée que le reste de l'infrastructure de test. Cela entraîne une consommation d'énergie plus élevée et une modification plus profonde de la conception du circuit intégré.

5 Sécurité de la Scan Encryption

La scan encryption offre une protection contre la plupart des attaques connues visant les infrastructures de test. D'une part, le chiffrement des données de test compromet la faisabilité des attaques basées sur le sniffage du canal de communication entre l'utilisateur et le dispositif cible. D'autre part, les attaques externes ne sont pas réalisables, car l'attaquant n'a pas de contrôle sur les données qui sont envoyées à l'infrastructure de test, il ne peut donc pas comprendre les résultats produits par le dispositif cible et lus par l'infrastructure de test. La sécurité de la scan encryption s'articule autour de deux points clés : *confidentialité des données* et *authentification de l'utilisateur*. La confidentialité des données transmises est une caractéristique clé du chiffrement en lui-même. En effet, lorsque l'utilisateur et l'appareil échangent des données cryptées avec une clé secrète, personne d'autre n'est en mesure de comprendre les informations transmises. Cela signifie que toute tentative d'attaque faite par des entités malveillantes au sein du système, et connectées à l'infrastructure de test, ne peut aboutir. En outre, toutes les attaques basées sur l'observation des états internes du circuit sont rendues impossibles par le fait que tout ce qui est déplacé hors de l'infrastructure de test est chiffré. L'authentification de l'utilisateur est obtenue grâce au fait que la connaissance de la clé secrète est fondamentale pour pouvoir communiquer efficacement avec le dispositif. Si l'utilisateur ne connaît pas la clé que le dispositif utilise pour le déchiffrement, il ne pourra pas prévoir la forme que prendront les données entrantes une fois qu'elles seront arrivées dans l'infrastructure de test. Il est donc impossible de forcer un état spécifique sur les registres internes du circuit. La seule possibilité qui reste à l'utilisateur non autorisé est d'entrer des données aléatoires dans l'infrastructure de test.

La seule capacité qui reste à l'attaquant est la possibilité de déplacer les données à travers des pins de scan, qui sont déchiffrées en interne, les rendant ainsi imprévisibles pour l'utilisateur non autorisé. La plupart des modèles de menaces que nous avons présentés sont basés sur l'hypothèse que l'attaquant peut insérer des modèles spécifiques à l'intérieur de l'infrastructure de test. Cependant, nous avons identifié certains scénarios dans lesquels même l'insertion de bits

aléatoires à l'intérieur de l'infrastructure de test pourrait constituer une menace pour la sécurité si des contre-mesures appropriées ne sont pas mises en œuvre. C'est le cas de certains systèmes à base de microprocesseurs, où des modes opérationnels protégés sont activés et désactivés en modifiant la valeur d'un seul bit dans un registre. Dans ce cas, même l'insertion de valeurs aléatoires à l'intérieur des chaînes de scan interne pourrait conduire à une attaque, rendant ainsi la contre-mesure de scan encryption inefficace. La scan encryption permet à l'attaquant d'entrer des données aléatoires dans l'infrastructure de test. Cela est possible car quelles que soient les informations que l'attaquant tente d'entrer par le pin de scan-in, elles seront corrompues de manière imprévisible par le chiffrement d'entrée. Malheureusement, les techniques de scan encryption n'offrent en aucune façon la possibilité d'empêcher un accès non autorisé, il est donc nécessaire de comprendre si l'insertion de contenu aléatoire dans l'infrastructure de test peut constituer une menace. Cela se produit dans les cas où, pour réaliser l'attaque, il suffit de forcer, au travers des chaînes de scan, un nombre limité de bascules à une valeur spécifique. Si le reste de l'attaque peut être effectué en mode fonctionnel, la présence de la scan encryption n'est pas suffisante pour empêcher l'attaquant d'entrer, après un certain nombre de tentatives, la valeur souhaitée à l'intérieur du dispositif. Dans cette thèse, nous avons émis l'hypothèse d'un tel scénario, en utilisant un microprocesseur équipé d'un environnement d'exécution de confiance (en anglais Trusted Execution Environment ou TEE) comme dispositif victime. En fait, dans ce type de microprocesseur, nous avons un seul bit contenant la valeur binaire qui détermine l'activation du TEE.

Nous avons montré plusieurs mécanismes d'authentification des utilisateurs visant à empêcher l'utilisation de l'infrastructure de test par des utilisateurs malveillants. Nous avons vu que ces techniques sont principalement basées sur l'insertion d'une clé secrète ou d'un mot de passe, et sur des protocoles de défi-réponse. De notre côté, nous avons identifié une opportunité offerte par le chiffrement, dans le but d'obtenir une forme d'authentification de l'utilisateur qui va au-delà des techniques standards traditionnelles. Plus précisément, il est possible de s'appuyer sur le dispositif qui déchiffre passivement les données d'entrée à l'aide d'une clé secrète. Par conséquent, en supposant qu'un attaquant n'ait aucun contrôle sur le processus de déchiffrement, il

est possible d'imposer des règles de format que les données saisies doivent respecter pour être acceptées par le dispositif. Ce faisant, le dispositif est capable de reconnaître, après le déchiffrement, si les données ont été envoyées par un utilisateur en possession de la clé secrète, ou par un utilisateur malveillant qui tente d'entrer des données aléatoires. Dans cette thèse, nous présentons la technique de scan encryption avec contrôle de parité (en anglais Scan Encryption with Parity check ou SEP), afin de contrecarrer l'attaque par scan présentée. L'hypothèse clé qui sous-tend la faisabilité de l'attaque par scan proposée, est que les schémas de chiffrement symétrique manquent d'un mécanisme d'authentification, de sorte que le destinataire accepte toujours le message reçu. Avec la technique SEP, nous proposons d'améliorer la scan encryption en ajoutant une vérification légère de l'intégrité des données transférées dans l'infrastructure de test. L'idée de base consiste à encoder, avant le chiffrement, les vecteurs de test en texte clair avec un algorithme de codage connu du public. Le dispositif destinataire, après déchiffrement, décode les vecteurs de test obtenus et vérifie leur conformité avant de les appliquer à la logique du circuit. Dans la technique SEP, nous avons choisi d'utiliser un code de parité couplé à un chiffrement par blocs afin de mettre en œuvre ce schéma d'authentification. La sécurité de ce schéma repose sur l'hypothèse suivante: *un utilisateur non autorisé n'est pas capable de créer un texte chiffré, de sorte qu'une fois déchiffré, le texte en clair qui en résulte est conforme au format souhaité*. Dans la technique SEP, on a recours au calcul de code de parité sur chaque bloc de chiffrement. Le décodage de la parité est effectué en conjonction avec le mécanisme de déchiffrement de l'entrée du scan. Un attaquant qui ne connaît pas la clé de chiffrement n'est donc pas en mesure de produire des vecteurs de test chiffrés valides qui passent avec succès la vérification de parité après le déchiffrement. Dès que le module de déchiffrement détecte un mauvais bit de parité, il lève un drapeau à l'intérieur du circuit. Lorsque le chargement de la chaîne de scan est terminé et que l'attaquant fait passer le circuit en mode fonctionnel, la présence du drapeau fait passer le circuit en état de protection. La technique SEP garantit les exigences de sécurité suivantes:

1. Les utilisateurs non autorisés ne peuvent pas contrôler ou observer l'état interne du circuit. En héritant du chiffrement de la chaîne de scan, nous nous as-

surons que toutes les données qui sont extraites par l'infrastructure de test ne sont pas compréhensibles par l'attaquant. En outre, après le déchiffrement des données d'entrée, l'exactitude des bits de parité est vérifiée. Cela empêche l'attaquant d'insérer des vecteurs de test avec un contenu aléatoire dans la chaîne de scan ;

2. Les messages échangés par le biais de l'infrastructure de test ne doivent pas être compréhensibles par des tiers. Cette propriété est assurée par le chiffrement des données de test effectué par un chiffrement par blocs standard. Ainsi, la sécurité sémantique du chiffrement est garantie. Notamment, même si l'attaquant est capable d'interroger le chiffrement par blocs interne et de prendre plusieurs paires texte clair/texte chiffré (c'est-à-dire un *attaque à texte clair connu*), il n'est pas possible de trouver une corrélation entre eux, d'où la clé secrète.

6 Conclusions

La nécessité d'un test conscient de la sécurité est une question urgente pour tous les systèmes électroniques modernes. Les infrastructures de tests invasifs sont obligatoires pour garantir la production de circuits intégrés à faible coût. Dans la plupart des applications, il n'est plus possible d'imaginer un système sans accès TAP. Dans cette thèse, nous montrons que la sécurité est un problème majeur qui affecte toutes les infrastructures de test. Pour cette raison, le flot DfT devrait prendre en considération la sécurité dès les premières étapes du développement des circuits intégrés. Dans cette thèse, nous avons montré que de nombreuses contre-mesures existantes pour les infrastructures de test sont basées sur l'ajout de modules cryptographiques pour l'authentification des utilisateurs. La plupart de ces mécanismes d'authentification sont basés sur un matériel cryptographique complexe, représentant un coût insupportable en dehors de certains marchés de niche spécifiques. De la même manière que la DfT a connu un processus d'automatisation au cours des dernières décennies, qui a permis son utilisation massive dans tous les segments du marché, nous voulons souligner que la nécessité d'une DfT sécurisé devrait également connaître une telle diffusion. Bien qu'il y ait maintenant de nombreuses propositions dans la littérature dans ce sens, le scénario est

encore trop fragmenté, et composé de techniques qui offrent une protection partielle et/ou dont la sécurité est difficile à prouver. Nous avons identifié la *scan encryption* comme une technique très prometteuse pour sécuriser facilement les infrastructures de test. La scan encryption est basée sur l'ajout de matériel cryptographique dont le coût peut être maintenu marginal selon le besoin du concepteur. Son déploiement est extrêmement simple et facile à automatiser. Le concepteur de la DfT établit les clés secrètes qui sont utilisées par le dispositif, et l'ensemble du flot de test est chiffré en conséquence. Cela empêche à la fois l'accès non autorisé par des utilisateurs malveillants et la fuite d'informations dans le canal de communication.

Dans cette thèse, nous avons analysé en profondeur les techniques de scan encryption et essayé de résoudre les problèmes qui ont été identifiés. Dans un premier temps, nous avons inspecté les techniques de pointe, en identifiant deux catégories : la scan encryption *basée sur le chiffrement par flot* et *basée sur le chiffrement par blocs*. Alors que la scan encryption par blocs avait déjà fait l'objet d'une analyse minutieuse et d'une activité expérimentale étendue, la scan encryption par flot a été proposée dans de nombreux travaux, mais n'a jamais fait l'objet d'une analyse de sécurité approfondie. C'est pourquoi nous avons proposé de nouvelles implémentations de scan encryption basées sur le chiffrement par flot, corrigeant ainsi certains problèmes de sécurité que nous avons trouvés dans l'état de l'art. La scan encryption proposée est basé sur la génération aléatoire d'un IV différent au début de chaque communication entre l'utilisateur et l'appareil. Bien que cela implique des coûts de mise en œuvre plus élevés également pour la scan encryption par flot, les deux techniques ont été amenées au même niveau de sécurité et peuvent toutes deux être envisagées sans risque pour le déploiement d'une DfT sécurisé. Nous avons largement comparé la nouvelle méthode de scan encryption par flot et la méthode de scan encryption par blocs déjà existante. Nous avons mis en évidence les avantages et les inconvénients des deux solutions, afin de fournir une ligne directrice aux concepteurs qui souhaitent utiliser la scan encryption pour sécuriser leur DfT.

La scan encryption, tant dans sa mise en œuvre par flot que par blocs, est capable de contrecarrer la plupart des attaques impliquant des infrastructures de test. En fait, tant la con-

trôlabilité que l'observabilité des ressources internes sont considérablement entravées. L'attaquant n'a aucun contrôle sur l'opération de déchiffrement qui est effectuée au niveau du port d'entrée. Ainsi, sa seule capacité est d'insérer des données aléatoires à l'intérieur de l'infrastructure de test. D'autre part, le chiffrement des données garantit la confidentialité des données de test en dehors des limites du dispositif cible. Ainsi, toutes les tentatives de piratage et d'accès illégal à le pin de sortie de scan n'entraînent aucune menace. Malgré la sécurité étendue offerte par la scan encryption, son absence de mécanisme de contrôle de l'intégrité des données la rend potentiellement vulnérable aux accès non désirés. Dans cette thèse, nous avons exploré les scénarios possibles dans lesquels un utilisateur malveillant peut exploiter la capacité d'insérer des données aléatoires dans l'infrastructure de test pour réaliser une attaque. Nous avons identifié une catégorie d'attaques, dans laquelle l'attaquant exploite les chaînes de scan interne afin de forcer un nombre très limité de bascules à une valeur spécifique. Dans ce cas, la contre-mesure de scan encryption ne suffit pas pour éviter un scénario similaire. Nous avons proposé un exemple d'attaque visant l'environnement d'exécution de confiance d'un microprocesseur. Si l'attaquant règle correctement la mémoire du microprocesseur, il est possible de déclencher l'exécution d'un code malveillant par l'insertion de données aléatoires dans les chaînes de scan. Pour cette raison, nous avons proposé une amélioration de la scan encryption comprenant un mécanisme de contrôle d'intégrité très léger. Dans cette technique, nous proposons d'inclure une règle de format aux données de test en texte clair que le dispositif cible peut facilement vérifier après le déchiffrement. Si ces règles sont respectées, les données reçues sont valides, sinon, elles sont considérées comme des données malveillantes envoyées par un attaquant. La sécurité de cette technique repose sur le fait qu'un attaquant ne peut pas produire un message valide qui résulterait en un format correct après décryptage, sans connaître la clé de scan encryption. Nous avons mis en œuvre cette technique en ajoutant des bits de parité dans les données de test. Au prix d'un léger surcoût en termes de temps et de surface de test, la scan encryption obtient une propriété de sécurité qui la rend robuste contre tout type d'accès non autorisé.

Nous pensons que la scan encryption est une technique très prometteuse pour protéger les infrastructures de test. Ses

caractéristiques lui permettent de fournir une DfT consciente de la sécurité, même dans des appareils peu coûteux. Sa flexibilité et la variété des mises en œuvre sont des caractéristiques qui permettent d'adapter la contre-mesure à différents besoins de conception. Nous pensons également que le développement d'un standard de test conscient de la sécurité est inévitable à l'avenir, et la scan encryption pourrait être un point de départ prometteur pour raisonner à ce sujet.

Acknowledgments

I would like to thank some people that, in different ways, contributed to this thesis. These acknowledgments have been redacted in different languages, according to the context and the people they are addressed to.

Grazie a Giorgio che, oltre a essere il mio direttore di tesi, è stato in questi anni anche un prezioso amico. Anche se le circostanze ci hanno portato a vivere in due città diverse, non ha mai smesso di motivarmi e di spingermi a cercare di arrivare sempre più lontano.

Merci à Bruno et Marie-Lise, pour leur soutien et leur patience pendant ces années de thèse.

Merci à Sophie pour ses précieux conseils et pour avoir participé au jury de thèse.

Merci a Guy Gogniat et Jacques Fournier pour avoir accepté d'être rapporteurs de ma thèse et pour avoir fait partie du jury.

Thanks to Regis Leveugle and Bernd Becker, for agreeing to join the thesis committee.

Grazie alla mia famiglia. Grazie a mio papà e a mia mamma, per avermi sempre motivato a fare meglio e per avermi sostenuto e permesso di arrivare fino alla fine del mio lungo percorso di studi. Grazie a Martina, per essermi sempre vicino nonostante la distanza geografica. Grazie a Gennaro, per essere entrato nelle nostre vite proprio durante questi ultimi anni e aver portato tante soddisfazioni.

Grazie a Cecilia, mia compagna di vita, che quotidianamente partecipa alla realizzazione dei miei sogni.

Merci à Mathieu, mon premier copain de bureau, pour m'avoir aidé beaucoup pendant ma première année à Montpellier.

Thanks to all the colleagues and friends, Bastien, Clement, Linh, Safa and all the others at LIRMM. They made my days at work a lot easier and funnier.

Merci à Caroline, pour son aide inestimable dans l'organisation des missions et dans la gestion des démarches bureaucratiques.

Thanks to all the colleagues and friends from the scientific community. Thanks to anyone that has shared professional advice (and drinks) with me during the conferences.

Grazie a Ilaria, mia collega e inseparabile amica in questa mia avventura a Montpellier. Grazie anche per essere sempre d'ispirazione per il mio lavoro e per spingermi ad astrarre i tecnicismi della mia ricerca.

Grazie a Marcello, per essere un caro amico. Ogni volta che sento qualche difficoltà, so che rivolgendomi a lui posso sempre trovare un saggio consiglio. I primi tempi a Montpellier sarebbero stati molto più difficili senza il suo supporto e i suoi suggerimenti.

Thanks to all the friends that I have found in Montpellier, Alina, Fernanda, Francesco, Cloé, Katherine, Sara, Umberto. They made my life in Montpellier a great experience.

Grazie a Matteo, per avermi fatto scoprire il LIRMM e condotto verso questa bella esperienza.

Grazie a Dizer, per farmi sempre ricordare il lato creativo della vita.

Grazie a Ler, amico del liceo, ritrovato a Montpellier. Grazie per essermi sempre vicino.

Grazie ad Aleandro, mio carissimo amico. Nonostante la distanza che ci separa, riesco sempre a sentirti vicino.

Grazie a Claudio, perché senza di lui questo dottorato non sarebbe neanche cominciato.

This thesis has been redacted during the lockdown period due to the COVID-19 outbreak. I want to thank every person that is taking part in the collective effort we are doing to counteract this disease.

Contents

Abstract **i**

Resume **iii**

1 Test des Systèmes Numériques iv

2 Menaces pour la Sécurité des Infrastructures de Test vi

3 Contre-mesures de Sécurité pour les Infrastructures de Test viii

4 Scan Encryption Sécurisé Basée sur le Chiffrement par Flot xii

5 Sécurité de la Scan Encryption xvii

6 Conclusions xx

Acknowledgments **xxv**

Contents **xxvii**

List of Abbreviations **xxxv**

Introduction **xxxix**

1 Test of Digital Systems **1**

1.1 Testing in the IC Production Flow 2

1.2 Design-for-Testability 4

Scan Chains 4

1.3 IEEE Test Standards 7

JTAG 7

IEEE 1500 9

IJTAG 10

2 Security Threats in Test Infrastructures **13**

2.1 Differential Scan Attack on AES 14

2.2 Classification of Threats 17
External Threats 18
Internal Threats 25
Summary 29

3 Security Countermeasures for Test Infrastructures 31
3.1 Classification of Countermeasures 31
User Authentication 32
User Authentication for IJTAG Networks 37
Privilege Based Authentication 42
Data Confidentiality 43
Device Authentication 45
Data Integrity 47
Attack Detection 48
Summary 51
3.2 Scan Encryption 52
Symmetric Encryption 52
Testing with Encrypted Data 55
Block Based Scan Encryption 57
Stream Based Scan Encryption 59
Summary 60

4 Secure Stream Based Scan Encryption 61
4.1 Scan Encryption Vulnerability 61
4.2 New Secure Scan Encryption Solution 64
Secure Scan Chain 66
Secure JTAG 72
4.3 Stream vs Block based Scan Encryption 77
Summary 83

5 Scan Encryption Security **87**

5.1 Security analysis against different attacks from the state-of-the-art 88

5.2 Security Threat against Scan Encryption 90

Trusted Execution Environment 91

Working Principle 92

Attack Implementation 93

5.3 Scan Encryption with Parity Check 96

General Overview of the SEP Technique 97

SEP Architecture 98

Security Analysis 100

DfT Flow with SEP Insertion 101

Area and Test Time Overhead 102

Discussion 104

6 Conclusions **105**

6.1 Future Perspectives 107

Scientific Contributions **111**

Bibliography **115**

List of Figures

1.1 Finite state machine implemented by the TAP controller of the IEEE Std. 1149 (JTAG) [1]. 8

2.1 Block diagram representing AES operations 15

2.2 Vulnerabilities in test infrastructures can be originated by (i) an external threat, caused by an unauthorized user accessing the IC; (ii) an internal threat, caused by malicious hardware planted inside the IC. 18

2.3 Taxonomy of the known threats on standard test infrastructures. 19

3.1 Taxonomy of the existing security countermeasures for standard test infrastructures. 32

3.2 High-level architecture of a generic stream cipher 54

3.3 High-level architecture of a generic block cipher 55

3.4 Functional scheme of scan encryption 56

3.5 Scan encryption based on block ciphers [82] 57

4.1 Schematics of the proposed stream based scan encryption. A TRNG produces the IV that is used to seed the stream cipher, together with the secret key. 65

4.2 Schematics of the saw bow technique for isolating the TRNG when the die is still connected to the wafer. 66

4.3 Architecture of the stream based encryption based of a scan chain infrastructure. 67

4.4 Stream based scan encryption applied to multiple scan chains supporting test compaction. 71

4.5 High-level architecture of Secure JTAG 72

4.6 Finite State Machine controlling the initialization procedure 73

5.1 Functional scheme of a generic microprocessor system implementing a Trusted Execution Environment. The Security Bit (SB) in the Configuration Register (CR) determines if applications executed from the user-space memory can access the protected resources. 92
5.2 Functional scheme of the architecture under attack. The user-space unprotected memory is filled with *jump* instructions redirecting to the malware instructions, which access the protected resources if *SB=1*. 94
5.3 Schematics of the SEP architecture. SI and SO pins (in red) carry encrypted test data and responses. The dashed line represents the parity signal, which is asserted by the decryption unit to the control unit. 99
5.4 High-level schematic of the DEC and ENC units. Parity signals are not present in the ENC unit. 99

List of Tables

2.1 Correlation between AES partial results and their plaintext [29]. 17

3.1 Analysis of the protection granted from each countermeasure against the known threats 52

4.1 Cost of the submodules composing the proposed countermeasure 69
4.2 Benchmark ICs used to evaluate the implementation cost of the TRIVIUM based scan encryption 69
4.3 Cost of the scan chain encryption with the TRIVIUM stream cipher on the chosen benchmarks 70
4.4 Area cost of the proposed countermeasure compared to an unprotected JTAG wrapper 76
4.5 Area overhead of different block and stream based scan encryption techniques. 80
4.6 Test time overhead of different block and stream based scan encryption techniques. 80

5.1 Complexity of the scan attack performed on a secure microprocessor based on the MiniMIPS CPU 95
5.2 Complexity of the scan attack on TEE performed on a secure microprocessor based on the MiniMIPS CPU protected with the SEP technique 101
5.3 Benchmarks Characteristics 102
5.4 SEP Insertion Overhead 102

List of Abbreviations

3PIP Third-Party Intellectual Property

AD Attack Detector

AES Advanced Encryption Standard

AE Authenticated Encryption

ASIC Application Specific Integrated Circuit

ATE Automatic Test Equipment

ATPG Automatic Test Patterns Generator

BIST Built-In Self-Test

BSR Boundary Scan Register

CBC Cipher Block Chaining

CPU Central Processing Unit

CRP Challenge-Response Pair

CR Configuration Register

CU Control Unit

DEC DEcryption Cipher

DES Data Encryption Standard

DfT Design for Testability

DoS Denial of Service

DR Data Register

DUT Device Under Test

ECB Electronic Codebook

ECC Elliptic Curve Cryptography

ECDSA Elliptic Curve Digital Signature Algorithm

EDA Electronic Design Automation

ENC ENcryption Cipher

FeRAM Ferromagnetic RAM

FF Flip-flop

FPGA Field Programmable Gate Array

FSM Finite State Machine

HLSIB Honeypot Locking SIB

HMAC Hashed Message Authentication Code

IC Integrated Circuit

IEEE Institute for Electrical and Electronics Engineers

IJTAG Internal JTAG

IoT Internet of Things

IP Intellectual Property

IR Instruction Register

IV Initialization Value

JTAG Joint Test Action Group

LFSR Linear Feedback Shift Register

LSIB Locking SIB

MAC Message Authentication Code

NLFSR Non-Linear Feedback Shift Register

NVM Non-Volatile Memory

OCD On-Chip Debugging

OTP One-Time Pad

PRG Pseudo-Random Generator

PUF Physically Unclonable Function

RAM Random Access Memory

RSN Reconfigurable Network

SEP Scan Encryption with Parity check

SE Scan-Enable

SIB Segment Insertion Bit

SI Scan-In

SKMU Secret Key Management Unit

SLFSR Secure Linear Feedback Shift Register

SLSIB Switching Locking SIB

SoC System on Chip

SO Scan-Out

SSC Secure Scan Chain

SVM Support Vector Machine

TAP Test Access Port

TCK Test Clock

TDI Test Data Input

TDO Test Data Output

TDR Test Data Register

TEE Trusted Execution Environment

TMS Test Mode Signal

TRNG True Random Number Generator

TRST Test Reset

WBR Wrapper Boundary Register

WDR Wrapper Data Register

WIR Wrapper Instruction Register

WSP Wrapper Serial Port

Introduction

The test is a fundamental step in the development of modern integrated circuits (ICs). In order to be able to effectively test circuits with high internal complexity, it is necessary to design appropriate test infrastructures within them. While these infrastructures make lean and effective testing practices possible throughout the entire product life cycle, they also offer an unwanted security backdoor. For this reason, there is a need to design test infrastructures using a security-aware approach. Many existing techniques use cryptographic implementations to prevent unauthorized access, or to provide confidentiality and integrity to test data transmitted between the user and the device. Among all the existing countermeasures, one of the most promising is the *scan encryption*. This countermeasure is based on the encryption of test data, which guarantees the confidentiality of test data and does not allow unauthorized use of the test infrastructure. In this thesis, we extensively discuss scan encryption techniques.

The contributions made by this thesis can be summarized as follows:

1. We provide an **extensive review** of the state-of-the-art on the security of test infrastructures. We propose a new classification of the security threats and countermeasures. Based on this classification, we perform a comparison between the existing countermeasures, and we identify the scan encryption technique as a promising approach for securing test infrastructures.
2. We thoroughly analyze state-of-the-art scan encryption techniques, identifying two different approaches for the encryption of test data (i.e., stream cipher and block cipher encryption respectively). Furthermore, we point out a **vulnerability** affecting existing scan encryption implementations based on stream ciphers.
3. We propose **new scan encryption techniques** based on stream cipher encryption, overcoming the vulnerability present in the state-of-the-art implementations.

The proposed implementations target different test infrastructures. Their cost is evaluated in terms of area and test time.

4. We compare the proposed techniques based on stream cipher encryption, with the existing techniques based on block cipher encryption. We highlight pros and cons of both implementations, providing a **guideline** to allow designers to opt for the most suitable scan encryption technique for their needs.
5. We analyze the security of the discussed scan encryption techniques, and we identify a **category of attacks** that can be carried out on ICs protected with scan encryption. This vulnerability affects all scan encryption implementations and stems from a lack of data integrity check in the scan encryption mechanism.
6. We **enhance** the scan encryption, adding a lightweight integrity check mechanism based on parity codes. This improved version of scan encryption offers a complete protection against all known threats.

This thesis is organized as follows. In Chapter 1, we provide a background on testing, and we describe the existing test infrastructures that are the object of the whole thesis. In Chapter 2, we review the existing security threats on test infrastructures. In Chapter 3, we describe the state-of-the-art security countermeasures for the test infrastructure design. We present a new classification of the existing techniques, and we focus on scan encryption techniques. In Chapter 4, we highlight a vulnerability affecting state-of-the-art scan encryption techniques based on stream ciphers. Furthermore, we present new scan encryption implementations based on stream ciphers overcoming the presented vulnerability. Finally, we compare stream and block based scan encryption techniques. In Chapter 5, we discuss the security properties of scan encryption techniques. We highlight the consequences of the lack of data integrity check, presenting an attack model on some secure microprocessors. Finally, we propose an enhanced scan encryption providing data integrity check based on parity coding. In Chapter 6, we draw the conclusions.

Test of Digital Systems 1

The development of test techniques for integrated circuits goes hand in hand with the innovations in semiconductor technology. The deployment of Moore's Law has been possible thanks to the constant reduction of the transistors size. However, this fast development has always been coupled with the presence of fabrication defects that necessarily affect cutting-edge semiconductor technologies. For this reason, *digital testing* is a fundamental practice for detecting the presence of defects on the production output and classifying ICs into *good* and *faulty*. Companies face high costs for guaranteeing an efficient test flow, but this cannot be avoided. For this reason, test engineers make big efforts in order to guarantee maximum fault detection with minimum test cost. The basic test procedure consists in stimulating the *device under test* (DUT) with test patterns, which stimulate the internal logic of the IC and propagate any possible fault effect on the outputs. Thus, it is possible to detect the presence of a defect comparing the test responses with reference results derived from the specifications. The objective is maximizing the *fault coverage* (i.e., the percentage of faults that are detected out of the total possible faults) while minimizing the test time. However, generating test patterns and measuring the fault coverage are issues that are far from trivial. In order to measure the fault coverage, it is necessary to determine a *fault model*, i.e., a model that correlates the physical defects with a logical behavior that can be represented and simulated at design time. The most popular fault model is the *stuck-at* fault model, where faults are represented as constant binary values imposed on the circuit interconnections. Generating test patterns requires the usage of specific algorithms, called *automatic test patterns generators* (ATPG), which are able to process the circuit netlist and generate a set of test vectors (i.e., input patterns for test purposes) that are able to provide the highest fault coverage. ATPG algorithms work very well and efficiently on combinational logic, but this is not the case for sequential logic. For this reason, *test infrastructures* come into play. Designers must employ *design-for-testability* (DfT) practices for efficiently generating test vectors. These are additional modules that are added to the IC design, in

order to support the generation and the application of the test [7, 8] .

[7]: Wang et al. (2006), *VLSI Test Principles and Architectures*
[8]: Bushnell et al. (2002), *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*

In this chapter, we introduce the most common DfT techniques and the related test infrastructures, which play a crucial role in this thesis. At first, we describe the overall test flow and its configuration in the production model of a typical integrated circuit. Finally, we describe in more detail some test infrastructures that are implemented inside the device at design time allowing the overall feasibility of the test procedures.

1.1 Testing in the IC Production Flow

The main objective of IC design and manufacturing flow is maximizing the production *yield*, which is the percentage of valid IC samples coming out from the production plant. For this reason, it is fundamental to be able to distinguish between good and faulty circuits as soon as possible in the production flow. During manufacturing, the test is performed resorting to the *automatic test equipment* (ATE), a programmable machine that sends test stimuli to the DUT. When the test procedure is started, the device inputs start being stimulated with test vectors and the results present on the output signals are recorded. Finally, test responses are compared with the expected outputs stored inside the ATE memory. If any mismatch is observed between the obtained responses and the expected ones, this means that the DUT is faulty. When silicon dies are still attached to the wafer, they are already submitted to a first test, called *in-wafer test*. In this phase, special probes, called *flying probes*, are used to reach the pads of the dies and to stimulate the circuit with test vectors. Using flying probes, it is possible to test multiple dies simultaneously and save time. After being cut from the silicon wafer, only dies that passed the test are sent for packaging, the others are discarded. After IC packaging, another test is performed. This kind of test is traditionally called *post-manufacturing test* and it can be performed by the manufacturing company or outsourced to specialized companies in the testing domain [7, 8].

When the device is shipped to the customer, this undergoes other tests that can be done to verify the compliance of the device to the specifications. Moreover, the target device can

contain some special configuration registers that must be set by the customer. This is carried out through a special procedure that is called *silicon bring-up*. Through this configuration, it is possible to configure special instruments embedded inside the IC (i.e., voltage regulation, current metering, internal sensors) [9] .

[9]: Portolan (2019), 'Automated Testing Flow: the Present and the Future'

When the device is deployed in the field, its testing necessities do not deplete. For instance, it is important for companies to be able to test faulty devices returned by customers. This must be assured in order to provide assistance and repairing damaged systems, and for providing feedback that is useful to the vendor. In the case of ICs equipped with microprocessors, it is also important to guarantee debug capabilities that can be exploited by software developers. Another scenario where testing in the field is crucial involves all the applications regarding safety-critical systems. The most prominent example in this direction is offered by ICs for automotive applications. In this case, it is common that at the system power-on all the ICs of the system are quickly tested and the integrity of all the functionalities is checked. This allows the system to detect the presence of faults before starting the car. Moreover, automotive subsystems that require the highest reliability levels (e.g., engine control unit, assisted break system) also need to perform *on-line* tests all along their operational activity [10] . In this scenario, it is common that specific circuit modules are periodically stopped in order to perform an on-line self-test. The most crucial assets in automotive systems must perform on-line tests providing almost total fault coverage up to five times per second [11].

[10]: Psarakis et al. (2010), 'Microprocessor Software-Based Self-Testing'

All scenarios described in this section show that test is a fundamental element of the IC development flow. From the very beginning, when silicon dies are still attached to their wafer, up to the last moments of operation in the field. All these test phases have the same objective of detecting possible faults in the circuit logic that could lead the system to a malfunction. They also share the same constraint on the test time, which must be as short as possible. In the next section, we describe more thoroughly how tests are generated and which are the kind of infrastructures included in the devices that are needed in order to support testability.

1.2 Design-for-Testability

Test generation is based on ATPG algorithms. These algorithms take as input the circuit netlist and the fault model. After a structural analysis of the netlist, a *fault list* is determined, containing all the possible faults that could affect the circuit. For each fault in the fault list, the ATPG searches for an input pattern (i.e., test vector) that is able to stimulate the target fault and to propagate its effect on the circuit output. The final response of the ATPG is the smallest subset of test vectors that is able to detect all the faults. A fault is *testable* only if it is both *controllable* from the input signals and *observable* on the output signals. Thus, the overall testability of a circuit (i.e., the easiness for the ATPG to generate test vectors) depends on the controllability and observability on the internal logic that is achievable acting on the primary inputs. In the case of combinational circuits, ATPG algorithms are capable of efficiently computing test vectors providing a very high fault coverage. On the other hand, this is a lot more difficult for sequential circuits. In fact, for testing a fault in sequential circuits, it is often necessary to run the circuit for several clock cycles, in order to drive the circuit into a specific state where the internal registers contain the correct values that are needed for stimulating the fault. After that, other several clock cycles can be needed in order to reach a state where the perturbation induced by the fault is observable on the circuit primary outputs. This problem leads, in the fortunate cases, to the generation of very long test sequences [7, 8]. Unfortunately, it is common that running the ATPG on a medium-complexity sequential circuit, the algorithm is not able to find any solution in a reasonable amount of time. For this reason, it is necessary to directly act on the design of the circuit logic in order to facilitate the task of the ATPG. In few words, we need to insert a *backdoor* on sequential circuits that allows the testers to transform the circuit into a combinational one.

Scan Chains

The most common DfT technique that is used for testing sequential circuits is the *scan chain* insertion. In this scenario, the flip-flops of the circuit are replaced by *scan flip-flops*. These flip-flops have an additional input, called *scan-in*, that

is connected to the output of another flip-flop of the circuit. The result is that all the flip-flops in the circuit are connected serially, similarly to a long shift register, and they form what is called a *scan chain*. A *scan enable* signal switches all the flip-flop between *functional mode* (i.e., the circuit is working normally and the flip-flops operate in the traditional way) and *test mode* (i.e., all flip-flops are configured in the scan mode, and they are disconnected from the combinational logic). The first and the last flip-flops of the scan chain are connected to the external pins of the IC through the *scan-in* and the *scan-out* pins respectively. This architecture allows the tester to directly shift values into the internal registers of the circuit using the scan-in access. Similarly, the scan-out pin allows the tester to shift out the content of the internal registers. This enables a test procedure that is carried out in the following steps:

1. The scan enable signal is set in order to switch the circuit to test mode.
2. Test vectors are shifted into the scan chain through the scan-in pin.
3. The circuit is switched back to functional mode for one or two clock cycles, according to the fault model that is used. In this phase, the values that have been imposed on the internal flip-flops are applied to stimulate the combinational logic. Thus, the test responses are captured in the internal flip-flops and on primary outputs.
4. The circuit is switched back to test mode and the content of the scan chain is shifted out through the scan-out pin. At the same time, the next test vector is shifted-in through the scan-in pin and the procedure restarts from point 2.

Let us suppose that a circuit has N flip-flops. The total test time is:

$$T = N(K + 1) + K \tag{1.1}$$

where K is the number of test vectors generated by the ATPG. It is evident that the length of the scan chain (i.e., the number of flip-flops of the circuit) strongly determines the test duration. This issue can be solved by spreading the circuit flip-flops over multiple scan chains. In this scenario, if N flip-flops are equally partioned into L scan chains, the

total test time becomes:

$$T = \frac{N}{L}(K+1) + K \tag{1.2}$$

In the case of complex circuits, where $N \gg K$, dividing the test infrastructure into a few scan chains can reduce the test time of one order of magnitude, which represents a strong gain in the production cost [7].

The major drawback of using multiple scan chains is the need of multiple scan-in and scan-out pins. This represents a serious problem in complex ICs, where the number of external pins is very reduced compared to the number of logic elements that are present in the netlist. This problem is solved using *test compaction* techniques based on `XOR` operations [12, 13] . At first, the ATPG generates test vectors for multiple scan chains. After that, each set of multiple test vectors is compacted in order to fit into a smaller number of bitstreams. The same operation is performed on the test responses. On the IC, a few scan-in pins are interfaced with a hardware decompressor that unravels the input bitstreams and generates the decompressed test vectors for the inner scan chains. Before the scan-out pin, test responses are compressed into a few bitstreams that are retrieved by the tester. Common hardware implementations of the compaction mechanisms are based on trees of `XOR` gates.

[12]: Barnhart et al. (2001), 'OPMISR: the foundation for compressed ATPG vectors'
[13]: Rajski et al. (2004), 'Embedded deterministic test'

Scan chains play a strategic role for the testability of digital logic, in all kinds of circuits. We have seen that scan chains have a very simple interface based on possibly two pins for scan-in and scan-out, and one pin for scan-enable. However, despite its easiness of management from an ATE point of view, this very simple interface has many limits. For instance, it is common that complex Systems on Chip (SoCs) are composed of several *intellectual property* (IP) cores supplied by third-party developers, whose netlist is already provided and equipped with scan chains. The IP core developers also provide test patterns related to their own sub-module to be used for post-manufacturing testing, and these are integrated into the test patterns targeting other parts of the SoC. In this scenario, it is not possible for the ATE to directly access the scan-in and scan-out pins of each internal IP core without the need of additional DfT structures. In the next section, we present the *test standards* that have been developed by the *Institute for Electrical and Electronics Engineers* (IEEE)

with the purpose of providing universal test interfaces and procedures.

1.3 IEEE Test Standards

The development of the IEEE test standards started in the 1980s with the Joint Test Action Group, abbreviated in JTAG, which started to develop the IEEE Std. 1149.1, published in 1991 [1] . The objective of the JTAG standard was the development of a test interface for integrated circuits that can be accessed when the devices are soldered on an electronic board. This work resulted in the development of the *Test Access Port* (TAP) mechanism, which represents today a universal interface for test infrastructures in integrated circuits. More recently, in 2005, the IEEE Std. 1500 was released [2] . This standard did not have the chance to be associated to a practical acronym, thus everyone still refers to it simply as IEEE 1500. This standard was developed in order to overcome some limitations of the JTAG standard in SoC applications. Finally, in 2014, the IJTAG (Internal JTAG) standard was released [3] . This standard introduces a new kind of test infrastructure that represents a real advancement with respect to the logic of the JTAG. The aim is to provide a flexible infrastructure for reaching different kinds of modules and embedded instruments inside SoCs. In this Section, we give some details on these test standards, highlighting the features that are essential for understanding the following chapters of this thesis.

[1]: Andrews (1991), 'IEEE Standard Boundary Scan 1149.1 An Introduction'

[2]: (2005), 'IEEE Standard Testability Method for Embedded Core-based Integrated Circuits'

[3]: (2014), 'IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device'

JTAG

The JTAG standard was primarily conceived for electronic boards testing. The key elements of this standard are the TAP controller, the *Instruction Register* (IR) and several *Data Registers* (DRs). The TAP controller is driven by an external signal and it executes an instruction, stored into the IR, selecting the DR that must be serially connected between the *Test Data Input* (TDI) and the *Test Data Output* (TDO) pins [14]. The TAP controller implements the *finite state machine* (FSM) that is depicted in Figure 1.1.

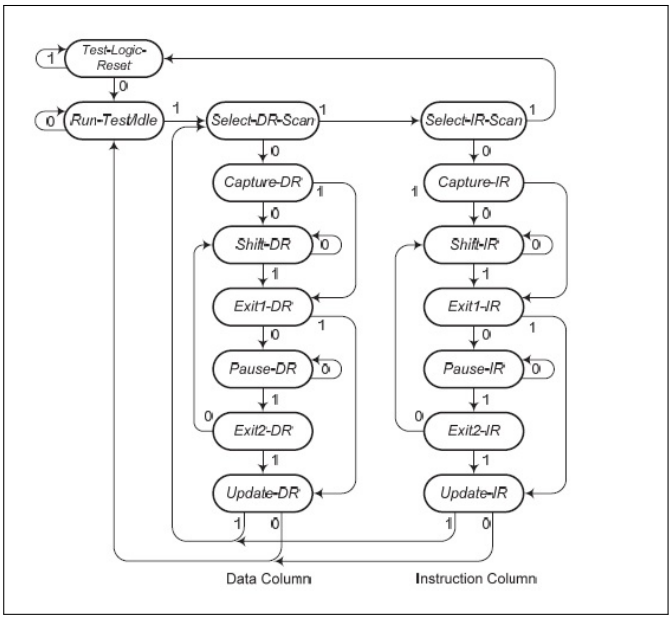

Figure 1.1: Finite state machine implemented by the TAP controller of the IEEE Std. 1149 (JTAG) [1].

The *Test Mode Signal* (TMS) is used to drive the TAP controller through all its states. The Test Clock (TCK) and Test Reset (TRST) are the other two signals of the TAP interface. In a typical test procedure, the user loads an instruction into the IR, through the TDI pin (i.e., `Shift-IR` instruction). An internal decoder accordingly selects the DR that must be connected between the TDI and the TDO when the controller is in the `Select-DR` state. Data registers can be of different kinds, and they can be personalized according to the needs of the designer. However, there are a few DRs that are recommended and defined by the JTAG standard:

- the **bypass register** is a single flip-flop that is interposed between the TDI and the TDO pin. This is selcted when data must be shifted through the device without any operation performed.
- the **IDCODE register** contains the device ID. This register cannot be overwritten, but the user can only shift its content out.
- the **boundary scan register (BSR)** is a special DR that is made of *boundary scan cells* located on the internal pins of the device. This allows the tester to serially access device pins and shifting values in and out of them.

The presence of the BSR enables the execution of the `INTEST` and `EXTEST` instructions. The `EXTEST` instruction is conceived to test the metal lines of the electronic board where the device

is soldered on. The `INTEST` instruction is fundamental for testing purposes. In fact, it is possible to exploit the presence of the BSR to feed data inside the internal scan chains. When multiple devices with a TAP controller are mounted on the same electronic board, there are different choices that can be made in order to interface them. The objective is having a TAP port at board level that allows the access to the JTAG infrastructures of all the devices on the board. Thus, it is very practical to configure the devices in a daisy-chain connection, where the TDO pin of a device is connected to the TDI pin of the next one. This way, it is possible to build a serial scan connection that traverses all the ICs on the system. Therefore, when test data are sent to one IC, they are shifted through all the other devices connected to the same JTAG infrastructure.

IEEE 1500

One limitation of the JTAG standard consists in its purely serial interface. If a user needs to test a device, test vectors must be integrated inside the serial JTAG interface. This means that a test vector, targeting L parallel scan chains, must be shifted into the device one bit at a time through the TDI pin. The same must be done for the test responses, shifted out through the TDO pin. This represented a problem, mostly for SoCs, where the presence of several internal IP cores, made the test access capability of the JTAG interface too limited. For this reason, the IEEE Std. 1500 was specifically developed for testing embedded cores inside SoCs. The main novelty of this standard is the presence of parallel *test input* and *test output* ports, which allow parallel access to the core under test. Thus, test vectors can be sent to the target core, provided by external pins or from other IP cores in functional mode. The other elements of the IEEE 1500 standard are inherited from the JTAG. Each IP core compliant with this standard is equipped with a test wrapper that is accessed through the *wrapper serial port* (WSP). The WSP can be serially accessed from external pins or from a SoC level JTAG infrastructure. Through the WSP, it is possible to access a *wrapper instruction register* (WIR) and some *wrapper data registers* (WDR). Amongst the WDRs, we find the *wrapper boundary register* (WBR), which has the same function of the boundary scan register in the JTAG standard [2]. It is evident that the IEEE 1500 standard does

not represent a radical change with respect to the logic of the JTAG. However, the main innovation is represented by the standardization of a test wrapper that can be implemented by IP core designers and integrated inside a wider test infrastructure. In fact, at SoC level, it is common to have a TAP interface that is connected to a board level JTAG infrastructure. Inside the SoC level JTAG infrastructure, the SoC integrator can introduce custom instructions that enable the access to the WSP of specific IP cores for testing purposes. This way, IP cores equipped with an IEEE 1500 compliant test wrappers can be perfectly integrated inside a higher level JTAG infrastructure and accessed from the board level TAP interface.

IJTAG

With the rising in complexity of SoCs, the traditional JTAG infrastructure became a bottleneck for test procedures. In fact, numerous sub-modules in a typical SoC need access from the external for testing and configuration purposes. These modules can be IP cores with a dedicated test procedure or embedded instruments that must be configured during silicon bring-up or in the field. When the number of these embedded instruments reaches the order of hundreds, or even thousands, connecting them all to the same TAP controller using the JTAG standard is impossible. In that case, all the embedded instruments and IP cores are serially connected in a daisy-chain connection. This implies that each operation that must be done on one of these instruments would need to shift test data through all the other instruments, resulting in very long test times. The IEEE Std. 1687, known as *Internal JTAG* (IJTAG), has been developed in order to deal with this kind of problems. The main idea behind IJTAG is the introduction of a dynamic test network with configurable length, opposed to the fixed-length network of the traditional JTAG. The IJTAG standard introduced the concept of *Reconfigurable Scan Network* (RSN), which is composed of many *Test Data Registers* (TDR). Each TDR is a register that represents the interface of the target instrument. Once data are shifted into a TDR, its value is *updated* into the target instrument register. Each TDR can be serially connected to the TDI and TDO pin of the TAP interface. However, each TDR can be gated using a *Segment Insertion Bit* (SIB), which is a modified scan cell,

which can be configured in order to bypass or not the portion of RSN that is connected to it. A possible scenario is the presence of a SIB for each TDR. When all SIBs are closed, the RSN has minimal length, because it composed only of SIBs. When the user wants to access a specific TDR, it is necessary to open the corresponding SIB (i.e., updating the value '1' into it) and, at that point, the target TDR is connected into the RSN, whose length is increased. In another scenario, multiple TDRs can be gated by the same SIB. In this case, the set of TDRs gated by the same SIB must be accessed at the same time by the tester [3]. Exploiting the IJTAG standard and its RSN, it is possible to schedule the test in different ways, accessing different instruments in parallel or not, according to the tester choices. This allows a radical optimization of the test time at the cost of an increase in complexity of the test scheduling [15–19].

In light of what has been said about test infrastructures, it is clear that they facilitate access to a large number of resources within integrated circuits. Specifically, scan chains offer an augmented controllability and observability on the internal flip-flops of the circuits. Moreover, test standards offer a universal port that allows the access to a plethora of internal resources from the external pins. Test networks, which are typical in the standard test infrastructures, facilitate the sharing of test data between different resources and intellectual properties. As we will show in the next chapter, all these features make test infrastructures an issue for security and trust in integrated circuits.

Security Threats in Test Infrastructures

2

The fact that test infrastructures pose a security problem in the integrated circuits design became clear since the early days of hardware security. In the last years of the 20th century, cybersecurity experts started turning their attention to the hardware implementations of cryptographic functions. In 1999, P. Kocher et al. published a milestone paper that introduced the notion of *side-channel attack* [20] . It was shown that measuring and observing the power consumption profile of a running cryptographic IC could easily reveal its secret key. Thereafter, the same attack was proposed for a plethora of physical quantities (e.g., electromagnetic emissions, acoustic waveforms), so called *side-channels*, that can be exploited for the same purpose. This full-fledged revolution in the cybersecurity field unveiled the weakness of the existing hardware implementations, extending the security burden to IC designers. It was only a matter of time before the accessibility, provided by test infrastructures, started being considered a security threat, in the same way as side-channels [21] .

[20]: Kocher et al. (1999), 'Differential Power Analysis'

[21]: Hely et al. (2004), 'Scan design and secure chip [secure IC testing]'

The first attack exploiting the scan chains, called *differential scan attack*, was published in 2004 by B. Yang et al. [22] This attack and the others published in a first period, focused on exploiting scan chains to steal the secret key from cryptographic ICs. At the same time, in parallel with the development of the test standards, the pervasiveness of complex test infrastructures gave the opportunity to conceive a plethora of different attack scenarios. The access provided by the TAP interface and the debug port, the interconnection between different IP cores made possible by the RSNs in the IJTAG standard, were all regarded as fertile ground for different attack scenarios. Since the beginning, research on the security of test infrastructures stood out from the research on side-channel attacks. In fact, while side-channel attacks necessarily involve physical properties of the ICs, scan attacks are studied at the architectural level, independently of the physical implementation. For this reason, most of the achievements in the test security literature sit on the statement (often implicit) that side-channel attacks are out of the scope [23, 24] .

[22]: Bo Yang et al. (2004), 'Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard'

[23]: Da Rolt et al. (2014), 'Test Versus Security: Past and Present'

[24]: Valea et al. (2019), 'A Survey on Security Threats and Countermeasures in IEEE Test Standards'

In this chapter, we present an exhaustive summary of the known threats involving test infrastructures. We discuss vulnerabilities regarding test infrastructures of different complexities, from simple scan chains to complex test networks based on the IEEE test standards. A classification and a taxonomy of these threats is proposed as one of the contributions of this thesis. However, before this summary, we thoroughly describe the differential scan attack on the Advanced Encryption Standard (AES) [25] , which is important in order to understand many contributions of this thesis.

[25]: Daemen et al. (2002), *The Design of Rijndael*

2.1 Differential Scan Attack on AES

The first scan attacks, exploited the presence of scan chains inside crypto-processors. In security applications, it is common to require the hardware implementation of cryptographic primitives. These can be stand-alone ICs or IP cores integrated inside a SoC. One of the most common cryptographic primitives are *block ciphers*. They are used to encrypt and decrypt data for confidential communications and their hardware implementation is able to reach higher troughput with respect to software implementations. This is very useful, for instance, for circuits involved in telecommunication systems. Crypto-processors base their security on the presence of a secret key, which is stored inside a secure memory in the IC, and it is shared with the parties that are involved in the communication. One of the most common block ciphers is the Advanced Encryption Standard (AES), which has been widely implemented in hardware [26] .

[26]: Verbauwhede et al. (2003), 'Design and performance testing of a 2.29-GB/s Rijndael processor'

Similarly to all block ciphers, the AES encrypts plaintext data per blocks, which are processed for a certain number of clock cycles and then transformed into blocks of ciphertext. In the case of AES, the size of the processed blocks is 128 bits. Figure 2.1 represents the block diagram of a typical hardware implementation of AES. The secret key (whose size can be 128 or 256 bits) is loaded from the memory and it is processed by the *key scheduler*, in order to generate many 128-bit *round keys*. At the beginning of the operation, the input plaintext (a) is combined with the first round key with an `XOR` operation. After that, data is processed by a round operation which is repeated ten times, in order to guarantee the required security level. In each round, the

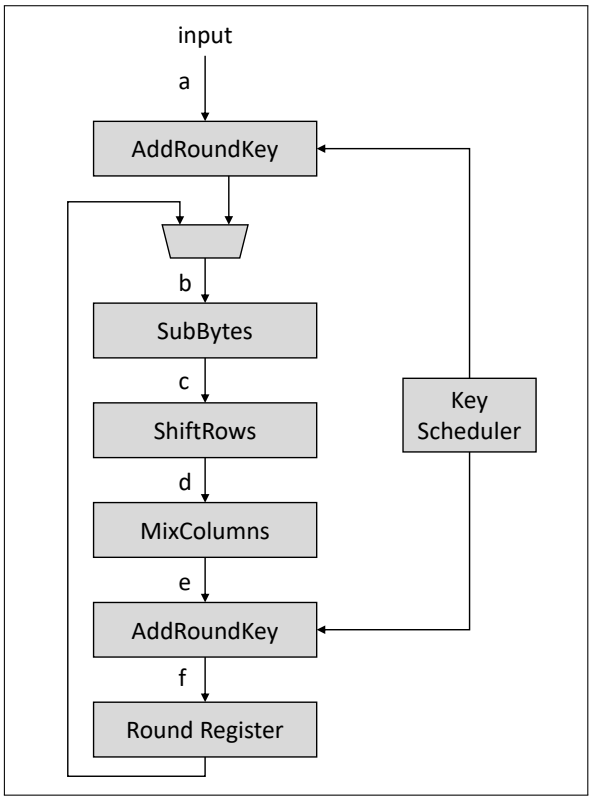

Figure 2.1: Block diagram representing AES operations

final `AddRoundKey` operation is performed with a different round key. The result of each round operation is stored inside a *round register*. After ten rounds, the value on the round register is the final ciphertext. In pipelined implementations, the same architecture from Figure 2.1 is unfolded into ten stages, each of them storing the result of the correspondent round inside a different round register [27] .

[27]: Mangard et al. (2003), 'A highly regular and scalable AES hardware architecture'

Round keys can be precomputed and stored into an on-chip RAM (*Random Access Memory*), or computed on-the-fly. The attackers are interested in retrieving the master key or one of the round keys. In fact, reversing the key scheduling algorithm is a trivial operation. the registers containing the master key, or the round keys, are included in the scan chains, they can be easily observed by the attacker. The attacker simply needs to switch the circuit to test mode, shift out the scan chain content and identify the flip-flops corresponding to the key registers. This has been the case in some satellite TV hacks that were popular in the past [28] . In 2006, B. Yang et al. published a paper showing that the secret key could be retrieved even if its registers were not directly accessible through the scan chains [29] . The attack is based on the differential analysis of the partial results that are stored into the round register after the first encryption round. Hence,

[28]: (), *Maestra Comprehensive Guide to Satellite TV Testing*

[29]: Yang et al. (2006), 'Secure Scan: A Design-for-Test Architecture for Crypto Chips'

this attack and all its derivatives were named *differential scan attacks*. The attack on AES published in 2006 was built upon the previously published attack on the simpler Data Encryption Standard (DES) [30] . However, we are going to focus on the AES attack, because this is still a widely implemented cipher in modern applications.

[30]: Standards (1977), *Data Encryption Standard*

Let us suppose that the attacker already knows the AES architecture that is implemented inside the crypto-processor. However, the structure of the scan chain is not known. Thus, as first thing, the attacker needs to know the exact position of the round register FFs (flip-flops) in the scan chain content. This is done by exploiting the following property: if we change one byte of the plaintext, this leads to the change of four bytes (i.e., 32 bits) on the partial result that is obtained after one encryption round. The attacker can send several input plaintexts, all equals except on one specific byte, and scan out the respective response from the round register. Observing the bits that differs between all the responses, it is possible to identify the position of the corresponding 32 FFs on the scan chain content. Simulations have shown that it is possible to retrieve the position of all 32 FFs using, on average, 6 patterns. Repeating this procedure for every group of 32 FFs in the round register, requires 24 patterns. At this point, the attacker can start the main phase of the attack and correlate the value of 32 FFs on the round register to one byte of the round key. The attack can be run with a very simple procedure. Different couples of plaintext patterns (a_1, a_2), having Hamming distance equal to 1 (i.e., $d_H(a_1, a_2) = 1$), are processed by the round function and their partial results are shifted out by the attacker. After that, the attacker computes the Hamming distance between the retrieved partial results (f_1, f_2). If the computed value is equal to one of the values from Table 2.1 on the facing page, it is possible to derive the input values of the `SubBytes` operation (i.e., (b_1, b_2)). Taking one between b_1 and b_2, it is possible to add it to the corresponding input value (i.e., a_1 or a_2) in order to derive the corresponding byte of the round key that is involved in the pre-round `AddRoundKey` operation. Statistically speaking, 32 patterns are needed, on average, in order to determine one byte of the round key. The same procedure can be repeated for each byte of the round key, giving a total of 512 patterns needed to retrieve the entire key. Adding the time required for the first phase, it is possible to prove that inserting 544 plaintexts and retrieving their partial encryption result

$d_H(f_1, f_2)$	9	12	23	24
(b_1, b_2)	(226, 227)	(242,243)	(122, 123)	(130, 131)

Table 2.1: Correlation between AES partial results and their plaintext [29].

through the scan chains, allows the attacker to completely derive the secret key, thus breaking the security of the whole system [22].

After AES, many other cryptographic circuits have been attacked exploiting some form of differential scan attack [31, 32]. In all cases, the feasibility of the attack is based on a similar procedure. The attacker runs the circuit in functional mode and, after a certain number of cycles, he/she switches it to test mode. The test mode offers the needed observability of the internal registers allowing the shifting out of their content through the scan chains. At this point, it is already possible to figure out some possible directions that could be taken, in order to build countermeasures against such attacks. One possibility consists in limiting the possibility for the attacker to switch between functional and test mode without being authenticated, or without losing the information that is stored inside the scan chains. A second possibility is to provide some form of confidentiality to the data that are shifted in and out the circuit in test mode. In the next chapters, we will better discuss this aspect. Differential scan attacks are an example of what can happen if unauthorized access to the scan chains is allowed in secure crypto-processors. However, test infrastructures are a security concern in any kind of IC, not only the ones targeting security applications. If we enlarge our vision to larger test infrastructures, such as the IEEE standards, it is possible to identify many more threats involving any kind of IC.

2.2 Classification of Threats

Design-for-Test goes decisively into contrast with the need for confidentiality and access restriction that is required in any kind of IC. The test infrastructure typically gives the user augmented controllability and observability on the internal details of the circuit. Moreover, the daisy-chain configuration, typical of the standard test infrastructures, ensures that multiple independent entities inside the system share the same data connection. This scenario can lead to serious

vulnerabilities, when sensitive data are shifted through the scan network. From a general point of view, the threats involving the test infrastructure of an IC belong to two main categories (Figure 2.2):

- *External threats*: unauthorized user that has the control over the TAP interface of the device.
- *Internal threats*: malicious device or IP core that is planted inside the system by a third-party entity (3PIP). In this case, the malicious device can access data propagated through the infrastructure it is connected to.

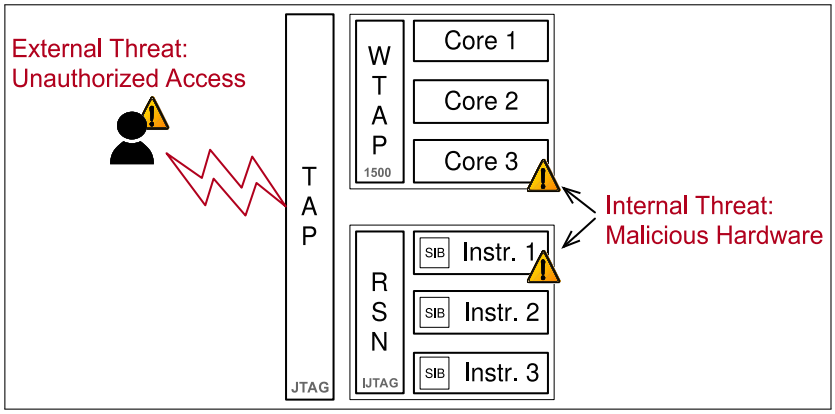

Figure 2.2: Vulnerabilities in test infrastructures can be originated by (i) an external threat, caused by an unauthorized user accessing the IC; (ii) an internal threat, caused by malicious hardware planted inside the IC.

Both external and internal attacks can be further classified into different categories, according to the kind of action that is performed by the attacker and the resources that are accessed (Figure 2.3). In this section, we will show this classification, and we will give some details about the known threats and the attacks belonging to each category.

External Threats

In this category of attacks, all the components of the system are supposed being trusted. However, if the attacker has physical access to the device, he/she is able to connect to the TAP interface and to exploit the highlighted controllability and observability on the device provided by the test infrastructure. Accessing the TAP interface it is possible to potentially reach every infrastructure that is connected to it. In the literature, we have identified three different targets that are exploited by the attackers:

- **scan chains**, which can be accessed from the TAP controller through the INTEST instruction;

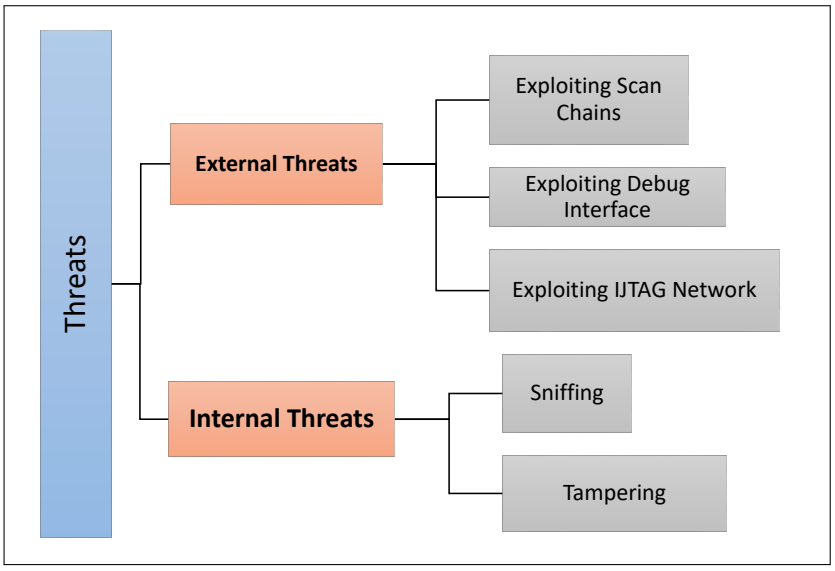

Figure 2.3: Taxonomy of the known threats on standard test infrastructures.

- the **debug infrastructure**, which is present on microprocessor-based systems for software development purposes;
- the **IJTAG reconfigurable network**, which is connected to the TAP controller and it is accessed through a dedicated instruction.

For this reason, we have classified external attacks in three sub-categories according to the kind of infrastructures that the attacker needs to access for succeeding.

Scan Chains

Internal scan chains of a target device can be accessed through the TAP controller. Executing some specific instructions on a JTAG infrastructure, such as the INTEST instruction, scan chains are connected to the TDI/TDO pins of the test interface. In another scenario, IP cores equipped with an IEEE 1500 test wrapper expose their internal scan chains to the TAP interface at SoC level. In both these cases, the attacker that has the control over the TAP controller is able to shift arbitrary values into the scan chains, and observe their content at any time through the TDO pin. This procedure is at the base of all the *scan attacks*, which aim to steal secret keys from crypto-processors. Another possible threat consists in using the scan chains in order to stimulate and observe the responses of the internal logic of the target device in order to perform *reverse engineering*.

Scan Attacks When crypto-processors are present inside a device, secret keys are usually stored inside integrated secure memories. Even if these secure memories are excluded from the scan chain insertion, scan attacks allow the attacker to retrieve the secret key. Several scan attacks have been proposed in the literature, targeting both symmetric and asymmetric cryptography implementations. We have already mentioned the most prominent attacks of this category, which targeted DES and AES block ciphers [22, 29] . These attacks, published by B. Yang et al. in 2004 and 2006 respectively, represent the beginning of a long series. In the following years, the same principle was applied to other kinds of crypto-processors, different from block ciphers.

[22]: Bo Yang et al. (2004), 'Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard'
[29]: Yang et al. (2006), 'Secure Scan: A Design-for-Test Architecture for Crypto Chips'

Y. Liu et al. presented an attack targeting stream ciphers based on Linear Feedback Shift Registers (LFSR) [31] . In this kind of ciphers, an LFSR produces a pseudo-random bitstream that is used to encrypt the plaintext. The objective of attackers is finding the internal structure of the LFSR (i.e., the implemented polynomial). The scan attack can be carried out running the LFSR in functional mode. After a certain number of clock cycles, the circuit is switched to test mode and the content of the LFSR is shifted out through the scan chains. The authors have brought to light some correlations between the internal states of the LFSR, that can be useful for the attacker to derive the structure of the LFSR and predict the generated keystream.

[31]: Liu et al. (2011), 'Scan-Based Attacks on Linear Feedback Shift Register Based Stream Ciphers'

The scan attack concept was extended to asymmetric cryptography as well. For instance, J. Da Rolt et al. presented a differential scan attack on Elliptic Curve Cryptography (ECC) [32] . The core of the computation in ECC crypto-processors is a point multiplication between the secret key and a scalar value. This operation is performed in several iterations, each involving a different portion of the key. Thus, the attacker can exploit the scan chain access to observe the intermediate results and derive the secret key.

[32]: Da Rolt et al. (2012), 'A scan-based attack on Elliptic Curve Cryptosystems in presence of industrial Design-for-Testability structures'

For some time after the publication of the first scan attacks, companies in the EDA (Electronic Design Automation) domain and their tools for the DfT automatic insertion were particularly affected. However, they were quick to reply that industrial DfT solution keep their security, because they always employ some form of compression/decompression, X-masking and X-tolerance, in order to deal with multiple

scan chains and complex test infrastructures. All these features were considered as a built-in protection mechanism, because the attacker was not able to directly observe the content of the scan chains, since they were processed by some compression circuitry before being shifted out. However, J. Da Rolt et al. [33] (further extensions in [34, 35]) extended the concept of scan attack to all kinds of test infrastructures, even the industrial DfT solutions that had been declared being secure by the EDA vendors. Successively, A. Das et al. [36] successfully implemented these attacks on real test infrastructures provided by the main EDA tools.

[33]: Da Rolt et al. (2011), 'New security threats against chips containing scan chain structures'

[36]: Das et al. (2013), 'Security Analysis of Industrial Test Compression Schemes'

All described scan attacks are possible as long as the attacker has the capability of switching the circuit from functional mode to test mode and vice versa without losing the computation state. For this reason, some countermeasures have been developed, based on the idea of resetting the scan chains when the circuit is switched from functional to test mode. To overcome this countermeasure, S. Ali et al. conceived a scan attack on AES that is entirely executed in test mode [37, 38] . In test mode, the AES plaintexts are loaded into the input registers through the scan chains. However, if the target crypto-processor completely disconnects the key (or uses a dummy test key) during test mode, this attack is ineffective.

[37]: Ali et al. (2013), 'New scan-based attack using only the test mode'
[38]: Ali et al. (2015), 'Novel Test-Mode-Only Scan Attack and Countermeasure for Compression-Based Scan Architectures'

Reverse Engineering The unauthorized access to the scan chains represents a threat not only for crypto-processors. In fact, any IC internals can be explored and reverse engineered exploiting the observability provided by the scan chain access. An attacker can set specific state values and retrieve the response of the combinational layers of the circuit. Hence, it is possible to build a database with stimuli/responses couples. A thorough analysis of these data allows the attacker to exactly reverse engineer the netlist of the circuit. Even if this scenario has been assumed as a threat model by many authors, only L. Azriel et al. in [39, 40] showed an implementation of the attack.

[39]: Azriel et al. (2017), 'Revealing On-Chip Proprietary Security Functions with Scan Side Channel Based Reverse Engineering'
[40]: Azriel et al. (2016), *Exploiting the Scan Side Channel for Reverse Engineering of a VLSI Device*

Debug Interface

Most modern ICs integrate a microprocessor and a debug infrastructure, which is essential in these systems. The debugging capability must be granted by the hardware designer, in

order to assist the software development process. The JTAG interface allows the user to access the debug infrastructure and perform On-Chip Debugging (OCD). If the debug interface is left accessible when the device is sent to the market, malicious users can exploit it. OCD tools allow the user to tamper the code execution at very low level. This means that security mechanisms implemented at software level can be overcome by OCD. Halting the software execution, a malicious debugger can modify and read the content of specific addresses of the memory, in order to cause unwanted behaviour in the system. All these operations can be easily performed using automated tools and high-level programming languages. In the literature, we have identified two kinds of attacks that stand on the possibility of performing OCD through the JTAG interface. The first category aims at performing the *memory dumping* of the system in order to clone it or steal valuable intellectual property. The second category uses debug access as an entry point to tamper the memory and obtain *privilege escalation* on the system.

Memory Dumping The first documented memory dumps relying on JTAG were performed by S. Willassen and M.F. Breeuwsma [41, 42] . The objective in both cases was to dump the whole content of a mobile phone memory for forensic purposes. In [42], the target device was a Nokia 5110 mobile phone. The author explains a detailed procedure, in order to access the external flash memory through the CPU (Central Processing Unit) JTAG controller and read all data out. However, the procedure presented in [41] is more comprehensive. It shows a more general attack that can be carried out on any portable device with JTAG access. A complete JTAG reverse engineering flow is presented, including the employed technique to find the TAP pins on the board. Once the JTAG infrastructure is accessed, the EXTEST or the DEBUG instructions are selected through the TAP controller. At this point, the attacker is able to send commands to the flash memory and dump all its content.

[41]: Breeuwsma (2006), 'Forensic imaging of embedded systems using JTAG (boundary-scan)'
[42]: Willassen (2005), 'Forensic Analysis of Mobile Phone Internal Memory'

F. Domke presented in [43] a reverse engineering procedure to explore undocumented JTAG instructions. Hardware manufacturers usually implement custom instructions in the TAP controller. They are meant for private in-house utilization, for this reason they are not referred in the device documentation. However, this paper shows a procedure that explores all

[43]: Domke (2009), 'Blackbox JTAG Reverse Engineering'

the undocumented instructions with a brute-force approach. The final result is that the attacker was able to find undocumented instructions that gave access to the scan chains and the internal bus, hence memory data could be read out.

JTAG related vulnerabilities have also affected high-security range devices. In [44] , S. Skorobogatov and C. Woods discovered a backdoor in a military chip. The victim device was an *Application Specific Integrated Circuit* (ASIC) from Microsemi, including a secure *Field Programmable Gate Array* (FPGA). The authors reverse engineered the JTAG infrastructure and found some undocumented instructions. Through these instructions, it was possible to download or overwrite the FPGA configuration, overcoming all the security features. Exploiting this backdoor, the ASIC producer could virtually retrieve all proprietary designs that their customers synthesized on their products.

[44]: Skorobogatov et al. (2012), 'Breakthrough Silicon Scanning Discovers Backdoor in Military Chip'

Privilege Escalation Penetration testers find serious vulnerabilities on consumer electronics devices on a day-to-day basis. In low-cost devices, producers keep the production costs to a minimum, necessarily scarifying the efforts for security. Some security companies publish on-line a selection of their most prominent attacks for advertisement purposes. For instance, the company in [45] published a JTAG attack performed on a very popular TPLink Router. The objective of this attack is gaining root access to the device. Once the JTAG interface is found, OCD allows the attacker to halt the execution of the bootloader at any moment. At this point, the memory can be conveniently tampered, in order to force the Linux kernel to run in Single User Mode, i.e., with root privileges.

[45]: Senrio (2018), *JTAG explained (finally!): Why "IoT" software security engineers and manufacturers should care*

F. Majéric et al. presented in [46] a JTAG attack exploiting a vulnerability of the Android kernel. Changing some specific values in the memory, it is possible to unlock the visualization of kernel modules addresses. Knowing the exact memory location of kernel modules is the starting point of several software attacks (i.e., buffer overflow). This vulnerability affected an Android build for Samsung Exynos SoCs and it was patched via software as soon as it was disclosed. However, the authors of this paper showed how, acting through OCD, it was still possible to perform the attack on these SoCs bypassing the software patch.

[46]: Majeric et al. (2016), 'JTAG Combined Attack - Another Approach for Fault Injection'

IJTAG Network

The attacker who manages to take control over the TAP controller of a SoC can also access the IJTAG reconfigurable network, if this is present. Since the IJTAG standard is relatively new, its application in commercial SoCs is still limited and not well documented. For this reason, we can find many attack scenarios in the literature, but there is still no knowledge of real implementations of these attacks. In these attack models, the target of the attacker is to access the configuration registers of specific instruments embedded in the target SoC. Since the design of the RSN is not known *a priori*, the attacker needs at first to reverse engineer it and figure out the arrangement of the SIBs. After that, the attacker can configure them in order to have access to the target instruments. Existing attack scenarios evaluate the kind of damage that can be done if certain kinds of instruments are accessed by malicious entities.

Hundreds of embedded instruments can be connected to a reconfigurable network. These instruments can be, for instance, BIST (Built-In Self Test) configuration registers. BIST circuits are used for on-line testing of specific IP cores. When the BIST is activated, it starts sending test stimuli to the target IP and it checks the test responses in order to determine the result of the test procedure. The tester can access the IJTAG RSN and writes the right value on the associated instrument to start the BIST procedure. While the BIST is running, the tester can configure the RSN in order to perform other tasks at the same time. In on-line testing applications, the *tester* is a circuit that schedules the test at the board level or at the SoC level. Since the BIST engines cause high power adsorption from the device under test, their activation must be carefully scheduled during the test phase [47] . If a malicious user is able to access the IJTAG network, many BIST engines can be activated at the same time and possibly cause overheating of the whole circuit. This scenario can lead to a Denial of Service (DoS) attack on the system. Even though any implementation of this attack has never been published, it has been mentioned as a threat in several publications. For example, in [48] the authors present an IJTAG security countermeasure as the main contribution of the paper, and they mention this attack scenario in order to justify the proposed countermeasure.

[47]: Zorian (1993), 'A distributed BIST control scheme for complex VLSI devices'

[48]: Dworak et al. (2013), 'Don't forget to lock your SIB: hiding instruments using P1687'

Embedded instruments connected to the IJTAG network also comprise SoC configuration registers. These configuration registers may be used to tune SoC parameters (e.g., internal voltage levels, clock frequencies) during the *silicon debug* phase, which is carried out by the hardware integrator after manufacturing. These configurations are part of the intellectual property of the SoC vendor. The authors of [48] mention this scenario as another possible menace affecting non-protected IJTAG infrastructures.

Internal Threats

The actual trend in the IC industry is the globalization of design and production. For this reason, the final products come from a production flow that involves many different companies. Fabless design companies usually provide proprietary IP cores to SoC integrators. In a typical design flow, the SoC integrator assembles all the IP cores, coming from different vendors, and designs the SoC level circuitry to grant the correct integration. In this phase, the SoC level testing infrastructure is inserted inside the design. The infrastructure is connected to the test interfaces of each IP core (e.g., TAP controller, IEEE 1500 test wrapper, BIST). The interaction between all these parties is of extreme importance for hardware security purposes. For instance, the SoC integrator does not necessarily have trust in the IP core vendors. Similarly, the IP core vendors does not have trust in each other. However, the IP cores are usually connected to the test infrastructure in a daisy-chain fashion. When the tester sends data to a target IP core through the TAP interface of the SoC, these are shared with other IP cores. The trust level of the SoC integrator with respect to the IP vendors can change according to different scenarios:

1. the IP cores are sold to the SoC integrator without test wrappers. In this case, the SoC integrator itself must wrap the IP cores with trusted test interfaces;
2. the IP cores are sold to the SoC integrator with test wrappers. In this case, the SoC integrator cannot equally trust the correct functionality of the IP cores, thus the test interfaces of the devices connected to the test infrastructure are considered untrusted.

The same considerations hold at board level, where ICs coming from different suppliers are mounted on the same board. In general, when test data are shifted through the test infrastructure of an untrusted IP core, there is no certainty that this will handle them according to predefined rules. Two possible threats have been envisaged in the literature so far:

- the untrusted IP core *sniffs* test data that are shifted through, and possibly steals sensitive information;
- the untrusted IP core *tampers* with test data while they are shifting through, and possibly corrupt their information.

As far as we know, there is no record in the literature of malicious devices that have been actually caught tampering with a test infrastructure. However, some authors have published several attack scenarios involving malicious devices exploiting their connection to the test infrastructure. These threat models have been largely used by researchers in order to motivate their countermeasures.

Sniffing

Each time a user wants to start a communication with a target device connected to the test infrastructure, he or she loads a certain instruction in the IR of the target. The other devices on the same network are programmed in BYPASS mode. A malicious device can be designed in order to store a copy of the data that are shifted through the bypass register. The stolen data can be used by the malicious device in different ways, according to the attack scenario. For instance, test data can be properly filtered in order to store the sensitive information into an internal memory, which is read by the attacker in a second moment. In more complex scenarios, the malicious device can even be able to send the data to a remote server. The attacker can process the collected data and retrieve sensitive information about the SoC and the other IP cores.

[4]: Rosenfeld et al. (2010), 'Attacks and Defenses for JTAG'

K. Rosenfeld and R. Karri presented in [4] a threat model involving malicious devices connected to a board level JTAG infrastructure. One of the presented attacks involves a sniffing device recording test vectors sent to another device connected to the same infrastructure. The malicious device must be

upstream the victim in order to make the attack successful. They state that illegally recording test vectors can leak confidential information on the design of the device under test. In a more complex scenario, two colluding devices, one upstream and one downstream the device under test, can record respectively test vectors and responses. This gives even more information about the internal logic structure of the victim.

Many boards on the market embed both a microprocessor and an FPGA, which is used to accelerate part of the computation. The FPGA is usually configured by downloading a bitstream through the JTAG interface. The content of the configuration bitstream consists in confidential information about the IP core implemented on the FPGA. Moreover, sensitive information about the internal structure of the FPGA is also contained inside the configuration bitstream. If the TAP port of the JTAG is connected to the same network where other devices are connected, the bitstream can be the target of a sniffing attack. In other cases, the configuration bitstream is stored into an external non-volatile memory and it is loaded into the FPGA at the system power-on. In this case, the content of this memory can be accessed through the JTAG interface of the system. It is possible to document the importance of this threat model in the technical documentation of FPGA manufacturers. For instance, Altera in [49] presents this kind of threat model, in order to motivate the importance of the security features that they implement on their FPGA.

[49]: Altera (2010), *Protecting the FPGA Design From Common Threats*

More recently, S. Kan et al. and R. Elnaggar et al. [6, 50] proposed a threat model involving malicious instruments connected to the IJTAG reconfigurable network. In this scenario, sniffing instruments can read out confidential configuration data that are shifted through the reconfigurable network.

[6]: Kan et al. (2016), 'Echeloned IJTAG data protection'
[50]: Elnaggar et al. (2018), 'Securing IJTAG against data-integrity attacks'

Tampering

Malicious devices connected to the test infrastructure can modify the content of the shifted data when they are in BYPASS mode. In the case of sniffing attacks, the behavior of the malicious device is completely passive. The sniffing action has no consequences on the behavior of the system. In the case of tampering devices, the behavior of the malicious device is the same as if it was set in BYPASS mode (i.e., it

shifts the values from the TDI pin to the TDO pin in one clock cycle). However, data that are shifted out the TDO pin are different with respect to the data entering the TDI pin. This kind of attack causes a different behavior of the system.

[4]: Rosenfeld et al. (2010), 'Attacks and Defenses for JTAG'

K. Rosenfeld and R. Karri described in [4] a possible data corruption scenario. If the tampering device is upstream the victim one, the data shifted into the victim can be corrupted. If the tampering device is able to smartly elaborate the modification of the data, it can lead the victim device to behave incorrectly. For example, the target device can be a microprocessor whose firmware is loaded via the test infrastructure. If the content of the firmware is modified while loading, it can be replaced with any malicious code that can cause a very different behaviour of the system.

Another hypothetical scenario presented in [4] can lead to a DoS attack involving a malicious device. The test infrastructure can be exploited to perform the on-line testing of an IP core inside a SoC. In this scenario, test vectors are stored into an internal memory. When the testing procedure starts, the test vectors are shifted through the test infrastructure and loaded into the DUT. When the responses of the DUT are ready, they are shifted out and compared with the golden ones. In an on-line testing scenario, the comparison is performed on-chip. If the tampering device is downstream the victim, corrupted responses can be delivered to the comparator. If the tampering device is properly programmed, test responses generated by the DUT can be modified into always being equal to the golden ones. At this point, if the DUT is faulty, the comparator is not able to detect it. This can lead to dangerous situations where the system goes into failure without the possibility for the system to detect it. However, the malicious device must know the test responses and the exact moment when the test is run, in order to successfully fake them.

[50]: Elnaggar et al. (2018), 'Securing IJTAG against data-integrity attacks'

The same principle can be exploited to threaten data integrity in IJTAG RSNs. In [50] , R. Elnaggar et al. presented a threat model involving malicious instruments connected to a reconfigurable network. Untrusted instruments are supposed to be capable of changing the value of specific bits that are shifted through their internal TDRs. This capability can be exploited in order to maliciously change the configuration of the RSN. A possible scenario is the following: a user starts a configuration session in order to include a set of instruments

in the RSN. During this process, the configuration bits are shifted through the malicious instrument, which changes the value of specific bits. The result is that an unwanted set of instruments is included in the RSN, without the user even realizing it.

Summary

The presence of test infrastructures makes ICs vulnerable to several kinds of attacks. In this chapter, we have classified all the threats that have been reported in the literature. We have noticed that some vulnerabilities are well-established in the scientific literature, and several attacks have been carried out. Other vulnerabilities identified in the literature involve trust issues that are more difficult to witness on commercial products, but they represent an equally important threat that must be addressed. For these reasons, IC designers are compelled to take security issues into account when dealing with test infrastructures of any kind. Starting from the classification proposed in this chapter, it is possible to derive two main objectives that should be pursued in order to obtain a security-aware DfT:

1. *authentication* mechanisms are necessary in order to avoid malicious users to access the test infrastructures and perform any kind of *external threat*;
2. data *confidentiality* and *integrity* are necessary to avoid *internal threats* based on malicious devices that sniff and/or tamper with data.

In the next chapter, we show the state-of-the-art countermeasures for securing test infrastructures, and we evaluate them relying on the security objectives outlined in this chapter.

Security Countermeasures for Test Infrastructures

3

When IC designers realized the risks caused by unprotected test infrastructures, the need for a security-aware DfT became pressing. At first, it was observed that the removal of the scan chains access could be an effective solution, because there could be no security risks if the scan chains were not connected to the IC pins. However, the scientific community quickly understood that this solution had a lot of drawbacks. First of all, disconnecting test infrastructures does not permit their exploitation after the manufacturing phase. This prevents any kind of usage of the test infrastructures in the field, strongly limiting the possible use cases. Even if it is possible to renounce to on-line test or debug functions for some specific applications, disconnecting scan chains poses another security risk that cannot be ignored. In fact, if the attacker has the capability of opening the IC package and probing the internal pads of the circuit, it is possible to bypass the countermeasure [51] . For these reasons, the scientific community started working hard to find secure design techniques for test infrastructures. The fundamental principle driving this kind of research is granting the presence and the usability of any kind of test infrastructure, making all known attacks impossible (or very hard) to perform.

[51]: Kömmerling et al. (1999), 'Design Principles for Tamper-resistant Smartcard Processors'

In this chapter, we present a summary of the most prominent countermeasures that are present in the literature. Our focus is on security techniques targeting complex test infrastructures based on the IEEE test standards. A classification and a taxonomy of these countermeasures is proposed as one of the contributions of this thesis. At the end of the chapter, we will better describe a particular countermeasure, called *scan encryption*, which will be the focus of the following contributions of this thesis.

3.1 Classification of Countermeasures

In Chapter 2, we have shown that threats affecting test infrastructures can be divided into *external* and *internal* threats.

In the literature, we can find proposals of countermeasures addressing both attack categories:

- external attacks are mainly thwarted by *user authentication* and *attack detection* techniques;
- internal attacks are counteracted granting *data confidentiality*, *data integrity* and *device authentication*.

We have divided these countermeasures into seven different categories (Figure 3.1). In this Section, we show the proposed classification, and we describe the most prominent state-of-the-art proposals.

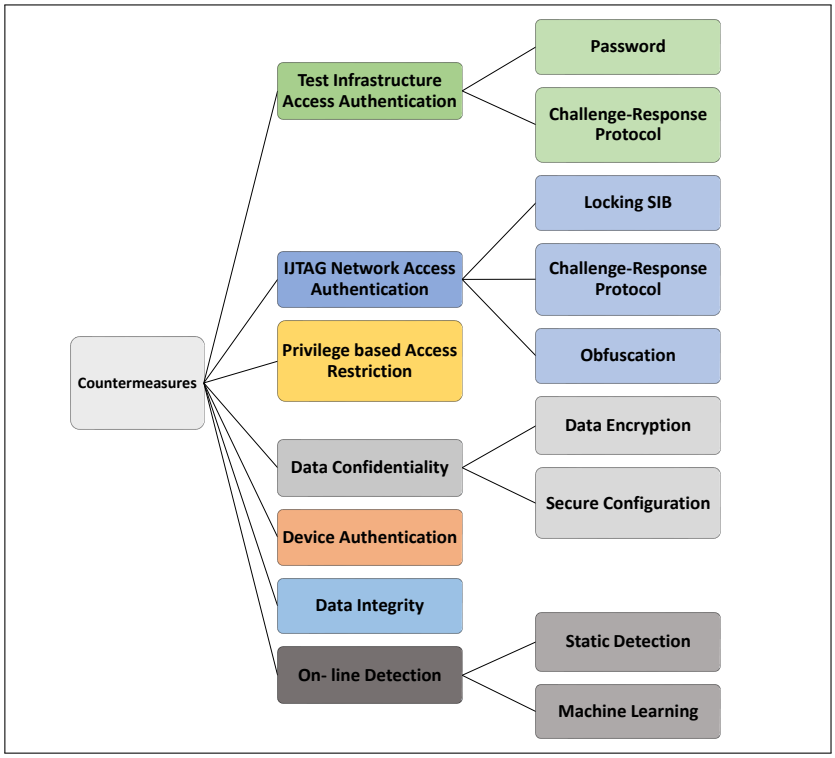

Figure 3.1: Taxonomy of the existing security countermeasures for standard test infrastructures.

User Authentication

This category of countermeasures aims avoiding unauthorized entities to access the test infrastructure. If the user is not authorized, the TAP controller is disabled and the JTAG instructions cannot be executed. This way, further access to the internal IP cores or to the IJTAG reconfigurable network is not possible. As a consequence, the exploitation of the internal scan chains or of the debug infrastructure is prevented. An authorized user is defined as someone that accesses the test infrastructure without causing any damage to all the parties involved in the development of the system.

Two categories of authentication techniques have been identified. One is based on the insertion of a *password* inside the TAP controller in order to lock or unlock it completely. The other category regroups a series of techniques based on *challenge-response protocols* and cryptographic primitives.

Password

Testing infrastructures protected by a password have their TAP controller locked by default. In the locked state, the execution of the JTAG instructions is not permitted, except from instructions that do not give access to sensitive data. In order to unlock the TAP controller, a secret password (or secret key) must be shifted inside a dedicated register. If the password is correct, the test infrastructure is unlocked and the authenticated user can access all its functionality. We recall two solutions based on this principle, one targeting the JTAG TAP controller of a generic device, the other targeting the IEEE 1500 test wrapper of an IP core integrated inside a SoC.

F. Novak and A. Biasizzo [52] presented a solution based on a modifies TAP controller. This TAP controller is able to execute two extra instructions, LOCK and UNLOCK. When the LOCK instruction is executed, the user must insert a password inside a specific register. When this operation is completed, the TAP controller is in a locked state. While in the locked state, the TAP controller decodes all the instructions into the BYPASS instruction. This condition does not allow the access to the test infrastructure. When the UNLOCK instruction is executed, the user is asked to insert the correct password. If the inserted password matches the one used to lock the system, the TAP controller is unlocked and full access is granted to the user.

[52]: Novak et al. (2006), 'Security Extension for IEEE Std 1149.1'

G. Chiu and J. Li [53] proposed a solution based on the integration of an LFSR inside the IEEE 1500 test wrapper. The polynomial implemented by the LFSR is secret. When the test infrastructure is in idle state, the test wrapper is locked. The user who wants to unlock the test wrapper must send a seed value to the LFSR. After that, the LFSR is run and a *golden key* is produced and stored inside the test wrapper. Then, the user shifts into the test wrapper a value that matches the golden key produced by the LFSR. If the value is correct, the test wrapper is unlocked. The efficacy of this countermeasure

[53]: Chiu et al. (2012), 'A Secure Test Wrapper Design Against Internal and Boundary Scan Attacks for Embedded Cores'

is based on the secrecy of the LFSR structure. Only the user that knows the LFSR polynomial is able to derive the right combination seed/key to unlock the test interface.

All the countermeasures of this category base their efficacy on the presence of a secret element that is shared between the target device and the authorized user. The efficacy of these methods can be easily jeopardized if the secret is leaked to unauthorized entities. It is possible to observe that in the solution [52] the secret can be dynamically changed programming each device with a different password. In this case, the leakage of one password undermines the security of only one specific device and it does not affect the whole production. The password can be changed every time the TAP controller is unlocked and relocked. This gives the possibility to the producer to easily change the protection password when needed. In the solution [53] the secret element is the structure of the LFSR. According to the design choices, each IP core can be sold with a different LFSR structure, in order to limit the damage in the case of leakage. However, if the LFSR structure of a specific IP core is leaked, this cannot be changed during the device lifetime.

Challenge-Response Protocol

In order to improve the security of password based techniques, more complex countermeasures have been proposed. These techniques are based on challenge-response protocols. The device sends a challenge to the user, which needs to prove his authenticity sending the right response back. The different countermeasures differ according to the kind of cryptographic primitive that is employed (e.g., *symmetric* or *asymmetric* cryptography) and the kind of tools that the user needs to authenticate. In the simplest cases, the user directly performs the challenge-response exchange with the device and personally holds the secret key that is required for the authentication. There are also more complex solutions where the user needs to obtain credentials from a secure server that holds the secrets required to compute the response. Only if the user successfully authenticates himself with the server, the latter computes the response and sends it to the device. This way the user does not need to directly hold the secrets and the risk of leakage is drastically decreased.

K. Park et al. [54] proposed an authentication protocol for the activation of the TAP controller. The protocol relies on symmetric cryptography and on the verification of credentials based on a secure server. The authentication procedure is based on two steps. In the first step, the user is authenticated by the server and obtains a credential that is stored into the device. An authentication between the server and the device is also performed. The device holds a secret key and the server stores a database with the key associated with each device ID. In the second step, the user asks the device for access. The access is granted only if both the user and the device hold a valid credential.

[54]: Park et al. (2010), 'JTAG Security System Based on Credentials'

CJ Clark [55] proposed an authentication protocol relying on the computation of the hash function of a random number. The challenge is a random number that is generated by the device. The challenge is sent to the user who appends a secret key to it. The resulting message is hashed with the SHA-256 algorithm. The result is sent back to the device, which verifies its validity computing the hash function itself resorting to the internally stored key. The authentication process is therefore based on the knowledge by the user of the secret key stored inside the device.

[55]: Clark (2010), 'Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments'

A. Das et al. [56] presented an authentication protocol based on Physical Unclonable Functions (PUFs) to protect the access to IEEE 1500 test wrappers. A PUF is a circuit element whose behavior depends on physical characteristics that are unique for each single device. Input challenges can be sent to the PUF, which gives unique responses as output. Nevertheless, the user must have access to a database containing all Challenge-Response Pairs (CRPs) of the target PUF. When the access to the target test wrapper is needed, the user must send a request to the device. At this point, the device sends a random value Δ to the user. The user searches through the CRP database for two responses having distance Δ. The two corresponding challenges are sent to the device, which are processed by the PUF. The device verifies if the produced responses have distance Δ. In the affirmative case, the access to the test wrapper is granted. The PUF must be queried at production time in order to create the CRP database. For this reason, a read-out system for the PUF response must be implemented and permanently disabled after manufacturing.

[56]: Das et al. (2012), 'PUF-based secure test wrapper design for cryptographic SoC testing'

R. Buskey and B. Frosik [57] proposed an authentication protocol based on asymmetric cryptography. In this solution,

[57]: Buskey et al. (2006), 'Protected JTAG'

the device holds the public key and a secure server holds the private key. The device sends a challenge to the server together with the device ID and the credentials of the user. The server checks if the user has the authorization for the requested operation. In the affirmative case, the server computes a response using the private key associated to the target device. This response is sent to the device. Finally, the device evaluates the authenticity of the response using its public key. The challenge-response protocol is based on ECC.

[58]: Das et al. (2013), 'Secure JTAG Implementation Using Schnorr Protocol'

A. Das, J. Da Rolt et al. [58] proposed another authentication protocol based on asymmetric cryptography. The utilization of the Schnorr protocol is proposed. This implements a signature algorithm based on ECC cryptography, called ECDSA (Elliptic Curve Digital Signature Algorithm). Using this primitive, an authentication scheme has been designed. The user and the device hold both a public key and a private key. At first, the user sends its public key and ID to the device. The device sends its ID and a challenge to the server in order to verify if the public key of the user is linked to a valid certificate. The server retrieves the public key associated with the device ID from its internal storage. After that, it creates a signature using the ECDSA scheme based on the public key itself, the ID and the challenge. This signature is then sent to the device that checks its validity. At this point, the device knows the public key of the user. The device then starts the Schnorr authentication procedure. In this exchange, the device, knowing the public key of the user, is able to verify the validity of its private key.

The key management is simpler for authentication protocols based on asymmetric cryptography. This is due to the fact that the public key does not need to be kept secret. In the solution [57], for instance, the device stores the public key, and there is no need to keep it secret. The only secret is the private key, which is stored into a secure server. The solutions based on symmetric cryptography have the advantage of a low implementation cost. For example, the solution [55] is based on hash functions. The implementation cost of hash functions is decisively lower than asymmetric cryptography, such as ECC.

User Authentication for IJTAG Networks

SoC designers can decide to apply different policies to access embedded instruments connected to the IJTAG reconfigurable network. This network is usually accessible through the TAP controller of the IC. When a specific JTAG instruction is executed, the RSN is connected between the TDI and TDO pins. However, the policy used to secure the access to the whole JTAG infrastructure can be different with respect to the one used for the IJTAG network. For instance, the designer could be interested in protecting only the IJTAG network, while there is no interest in limiting the access to the TAP controller. This is the case when the security is already provided by the IP cores integrated inside the SoC. In another possible scenario, the access to the IJTAG network is granted to a subset of the entities that have access to the TAP controller. In this case, the secret to access the TAP controller is handed out to authorized users, but only a subset of these users also owns the secret to access the IJTAG network. For this reason, several authentication mechanisms have been proposed in the literature to specifically secure the access to the IJTAG RSN. When an attacker tries to illegally access the RSN network, he/she does not know the exact structure of the RSN. At first, the attacker has to figure out the length of the network in its default configuration. After that, he or she tries to spot the SIBs and to open them, in order to progressively reverse engineer its structure.

Three categories of authentication techniques for IJTAG networks have been identified in the literature. The first one is based on *locking SIBs* that gate the access to private regions of the network. The knowledge of a secret password is necessary in order to open these SIBs. The second category is based on *challenge-response protocols* that enable the access to the network (or to smaller parts of it) only to authorized users. The last category aims to *obfuscate* the structure of the network, increasing the complexity of its reverse engineering.

Locking SIB

It is possible to restrict the access to specific instruments connected to the IJTAG network hiding them behind special SIBs. These SIBs are locked by default and only authorized

users, who know a specific secret, can open them. This prevents the attackers from accessing the embedded instruments connected to the RSN.

[48]: Dworak et al. (2013), 'Don't forget to lock your SIB: hiding instruments using P1687'

J. Dworak et al. [48] proposed a special SIB, called *Locking SIB* (LSIB). The LSIB is a modified SIB with additional logic that enables its opening only if a predefined value is sent to the additional key ports. The key is stored into add-on FFs in the RSN that are connected to the key ports of the target LSIB. When the right value is loaded into these FFs, the target LSIB is opened. The LSIBs protect segments of the network that can be accessed only by authorized users that know the secret key.

[59]: Liu et al. (2015), 'Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key'

H. Liu and V. Agrawal [59] proposed a different kind of LSIB that relies on a Secure LFSR (SLFSR) integrated in the scan network. The SLFSR is placed downstream the LSIB that must be protected. When the LSIB is closed, the SLFSR is activated and the data that are shifted through it are scrambled. This way, the attacker is confused while trying to reverse engineer the network. The parallel output of the SLFSR is connected to the key pins of the LSIB. When the right value is generated by the SLFSR, the LSIB is unlocked. Once the LSIB is unlocked, the SLFSR switches to a simple shift mode and data passing through the RSN are not perturbed anymore. The security of this solution relies on both the knowledge of the secret key to unlock the LSIB and the knowledge of the structure of the SLFSR that must generate the secret key.

[60]: Satheesh et al. (2016), 'Securing IEEE 1687 Standard On-chip Instrumentation Access Using PUF'

N. Satheesh et al. [60] presented a countermeasure in which the LSIB is unlocked resorting to a PUF-based security module. The security module receives a challenge from the user. This challenge is sent to the PUF, which generates a response. The response is compared with the output of an LFSR. If they are equal, the LSIB is opened. The LFSR produces the same output as the PUF when the user clocks it for n cycles. The value n is a secret that the authorized user must know. The secret n depends on the challenge given to the circuit. Since the PUF does not have any connection with the external pins of the IC, its CRPs cannot be collected at manufacturing time. The value n associated with each challenge is measured at post-manufacturing running the LFSR until the LSIB is unlocked. This system provides weak security, because its complexity needs to be kept low for the feasibility of the key determination procedure.

The techniques based on LSIBs presented in [48] and [59] rely on a secret password established at design time and hardwired inside the logic of the LSIB. In [61] , the authors show a side-channel attack targeting the LSIB technique. Moreover, the solution [59] also relies on an LFSR whose structure is established at design time. For this reason, any leak of this information undermines the security of all the samples that share the same design. The same can be said for the solution [60], even if the behavior of the PUF is different in each device. In any case, retrieving the secret for a single device is not hard, because the complexity of the attack is equal to the complexity of the procedure executed at design time for retrieving the device key.

[61]: Gupta et al. (2017), 'Mitigating simple power analysis attacks on LSIB key logic'

Challenge-Response Protocol

The IJTAG network can be also protected by an authentication module that implements a challenge-response protocol. According to the implementation, the authentication procedure can protect the whole RSN, a part of it, or a specific embedded instrument.

The solution proposed by CJ Clark in [55] can be applied to the IJTAG network or to a specific instrument connected to it. The authentication mechanism is the same as for the JTAG infrastructure (see Chapter 2). The difference is that each instrument owns a different key. However, the SHA-256 engine, used to verify the validity of the response, can be shared by all the instruments.

[55]: Clark (2010), 'Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments'

R. Baranowski et al. [62] presented an authentication protocol that gives the access to a secure region of the IJTAG network. In the first step of the protocol, the device sends a challenge to the user. The challenge is a random number produced by a True Random Number Generator (TRNG). The user concatenates the received challenge with the keys associated to the target instruments. After that, the resulting message is hashed to obtain the response that is sent back to the device. The device checks if the hash is valid; if this is the case, the user is authorized to access the target instruments. When the authentication is successful, the controller opens the section of the RSN containing the secured instruments.

[62]: Baranowski et al. (2015), 'Fine-Grained Access Management in Reconfigurable Scan Networks'

The techniques based on challenge-response protocols require each instrument to have a secret key associated with

it. Therefore, each instrument must manage its access rights independently. For example, in the solution [55], each instrument has to manage the verification of the response sent by the user. This means that each instrument has necessarily an area overhead due to the verification logic. In the solution [62], the authentication controller is centralized. Nonetheless, in this case it is necessary to guarantee a path in the RSN where only the accessible devices are connected. This leads to non-negligible routing issues.

RSN Obfuscation

Another way to secure the access to the IJTAG network is increasing the complexity of the exploration algorithms that are necessary to perform an attack. The attacker who does not know the design of the circuit tries to figure out the structure of the RSN in order to spot the position of the SIBs and open them to access the associated instruments. If the geometry of the RSN is unpredictable, the time required by an attacker to reverse engineer it increases considerably.

[48]: Dworak et al. (2013), 'Don't forget to lock your SIB: hiding instruments using P1687'

A couple of design techniques for complicating the structure of the RSN were presented in [48] . In order to make the attack more difficult, *trap bits* are introduced. These special cells are inserted in the RSN: if the wrong value is updated into them, the output of the cell gets irreversibly stuck to a fixed value until a global reset is issued. Trap bits can be connected to a key input of the LSIB to prevent the attacker from unlocking them even if the right key is set in the key bits. Another solution is to use the trap bits to activate an alternative path, in order to put the key bits of the LSIB out of the scan path. This way it is impossible for the attacker to continue forcing the key without a reset of the whole circuit. Trap bits can also be used independently of the LSIBs. For instance, it is possible to connect them to simple SIBs in order to force their closure. Alternatively, they can be set in order to block the shifting of the RSN. Another technique proposed in the same paper is the implementation of *hierarchical locks*. They are structures where the key bits are spread on multiple levels of the network. For this reason, it is necessary to unlock specific LSIBs before being able to access all the key bits that are needed to unlock the target LSIB.

[63]: Zygmontowicz et al. (2014), 'Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687 network'

A. Zygmontowicz et al. proposed in [63] other techniques to combine with the LSIBs. The first one is the introduction

of special LSIBs, called *honeypots* (HLSIB). HLSIBs provide access to a sub-network that does not contain any instruments. Instead, a target LSIB is disabled as far as the HLSIB stays open. This gives a fake feedback to the attacker, who may think to have successfully opened the LSIB. In this case, the attacker is motivated to explore the sub-network opened by the HLSIB without knowing that this has no instruments connected (i.e., a honeypot). The second proposal consists in creating a network where some LSIBs are opened by default, and they need to be closed in order to be able to open other LSIBs. This should confuse the attacker who does not expect to have to reduce the length of the network in order to completely open it. The third proposal consists in introducing *switching LSIBs* (SLSIB) that open different networks according to the value that is updated into them. One of the hidden networks is a dead end, the other gives access to the protected instrument. If both the networks have the same length, the attacker does not realize that an LSIB was there because the length of the network does not change.

S. Kan et al. proposed in [6] a technique that gives the possibility to program the geometry of the RSN at post-manufacturing. The authors proposed the insertion of *stub chains*. They are additional portions of the scan network that can have different lengths. The configuration of the stub chains is set selecting multiplexers that convey the scan flow on stubs of different lengths. This configuration is decided at manufacturing time using fuses or PUFs. This way, each sample of the device has a different configuration of the stub chains. Therefore, the attacker who is able to reverse engineer the RSN of one device is not able to perform the same attack on all the others.

[6]: Kan et al. (2016), 'Echeloned IJTAG data protection'

In the solutions [48] and [63] the countermeasure is coupled with the use of the LSIBs. The time required to open an LSIB with a brute force attack is proportional to 2^n, where n is the number of bits of the secret key. These techniques aim at increasing the attack time. In the solution [6], the structure of the RSN is simply made unpredictable because it is different on each sample of the target device.

Privilege Based Authentication

The countermeasures grouped in this category are an extension of the user authentication techniques. In this case, all the users do not have the same kind of authentication, but they get different privileges on the testing infrastructure according to the trust level they have.

[55]: Clark (2010), 'Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments'

The authentication protocol presented in [55] allows the system to provide different authentications according to the group of JTAG instructions that can be executed. For example, each set of private instructions can be associated with a different key. This way, the users are authorized to use a set of instructions by knowing the associated keys.

[64]: Backer et al. (2015), 'Secure design-for-debug for Systems-on-Chip'

J. Backer et al. presented in [64] an authentication mechanism for the debug infrastructure. The debug infrastructure is accessed resorting to the JTAG port of the IC. The aim is to filter out sensitive assets that can be retrieved from the system in debug mode. Each asset has a tag, linking it to its owner. The user must be authenticated to access the debug infrastructure. At the end of the authentication procedure, a *privilege level* is assigned to the user. Each asset that is read from the system is checked at runtime to verify that the privilege level of the user allows him/her to be an owner of that asset.

[65]: Pierce et al. (2013), 'Enhanced Secure Architecture for Joint Action Test Group Systems'

L. Pierce and S. Tragoudas presented in [65] a technique based on a module that manages the authentication protocol and gives the user a *privilege level*. Moreover, an access monitor filters the update signal of the boundary scan cells. The *access level* of each resource is stored into a memory. When the resource is accessed, its access level is compared with the actual privilege level of the user. The update signal is forwarded only if the access level of the resource is less or equal than the privilege level of the user.

[62]: Baranowski et al. (2015), 'Fine-Grained Access Management in Reconfigurable Scan Networks'

The solution proposed in [62] expects each instrument connected to the IJTAG network to have a secret key, which is used for the user's authentication. At the end of the authentication procedure, the user can access only the instruments that he/she is authenticated for. In order to guarantee this condition, a Secure Scan Chain (SSC) is activated. The SSC only connects the instruments for which the user has been

authenticated. The other instruments are connected to another portion of the RSN that is not physically reachable by the SSC.

The solutions proposed in [55] and [65] target the JTAG infrastructure. In [55], the privilege level of the user determines the kind of JTAG instructions that can be executed, regardless the content of the accessed data. In [65], the user having access to the TAP controller can execute any instruction. However, the content of the accessed data is checked. For example, two users having different privilege levels can both perform debugging, but only one of them may be allowed to access a range of memory locations containing confidential data. The solution [64] is specifically related to the debug infrastructure, which is accessed by the JTAG interface. In the solution [62] the instruments in the IJTAG network must be grouped in different chains, each one accessible only by the users having some specific privileges. The user that wants to obtain the privilege to access a specific SSC must know the secret keys of all the instruments attached to it. In the case in which an instrument belongs to more than one privilege group, it must be necessarily reached by more than one SSC. This may cause non-negligible routing issues.

Data Confidentiality

When sensitive data are exchanged between the user and the device, the possibility of sniffing from a third malicious entity represents a threat. This risk is present both in a board environment and inside a SoC, where the malicious entity is an internal IP core. In addition, the IJTAG networks need to be protected when confidential data could be shifted through embedded instruments that are not trusted. Countermeasures to provide data confidentiality have been largely developed by researchers.

We have identified two categories of countermeasures that provide confidentiality of test data. The first category relies on the encryption of the test patterns and it can be applied to all kinds of test infrastructures. These are called *scan encryption* techniques. The second category is more oriented to protect the IJTAG networks. The configuration of the RSN is properly modified in order to *isolate* the untrusted instruments when confidential data are shifted through it.

Scan Encryption

In this thesis, we will present new security techniques for test infrastructures that belong to the category of scan encryption. For this reason, we will detail the state-of-the-art about scan encryption in the following section. Here, we give a short introduction about the security properties of this technique.

Scan encryption techniques are based on the deployment of encryption primitives to encrypt all data that is involved in the test infrastructure. The IC designer must set up a secret key, which is securely stored inside the device. After that, the designer must encrypt the test data using the secret key. The encrypted data are sent to the target device, which decrypts them, processes them and produces encrypted responses for the user. Here, we point out the security properties that are granted from an encrypted communication with the device. In fact, this approach leads to a twofold protection:

1. data confidentiality is provided, since no unauthorized entity can understand the content of the communication between the user and the test infrastructure;
2. user authentication is partially provided because unauthorized users, which do not know the secret key, do not have controllability on data that is inserted through the TDI pin.

It is possible to appreciate how scan encryption potentially covers a greater number of threat scenarios than already mentioned user authentication techniques.

Secure Configuration

When dealing with IJTAG networks, it is possible to exploit the reconfiguration capability of the RSN not including untrusted instruments when a confidential communication is carried out.

[62]: Baranowski et al. (2015), 'Fine-Grained Access Management in Reconfigurable Scan Networks'

The countermeasure proposed in [62] allows the designer to keep the untrusted instruments away from the secure scan chains. If an instrument deals with confidential data, it can be connected to a secure scan chain. This way, the user is sure that sensitive data are not shifted through untrusted devices. Only authenticated users have access to the secure

scan chains. Further developments have been proposed in [66, 67]

M. Kochte et al. proposed in [68] a design technique for secure IJTAG networks. Secure access patterns can be generated, such that untrusted instruments are not involved in the network when confidential data are present in the communication. The secure patterns configure the network in order to keep untrusted instruments isolated. When this configuration cannot be achieved due to the structure of the network, a modification of the design is performed and a bypass segment is added in order to redirect the data flow. When confidential data are shifted through the network, the bypass segments are activated and the data are not shifted through the untrusted instruments.

[68]: Kochte et al. (2017), 'Trustworthy reconfigurable access to on-chip infrastructure'

The solution [62] is very efficient when the untrusted instruments do not belong to the set of devices that need the user authentication. In this case, they are not part of the secure scan chain. Thus the confidential data, which are shifted through the secure scan chain, are not exposed to the untrusted instruments. The solution [68] is more versatile, because any instrument considered untrusted can be isolated from the confidential data. The main limitation is that this technique is applied at design time, without the possibility to update the security policies at a later stage. Moreover, the insertion of bypass segments does not avoid the possibility to electrically leak the confidential data on the untrusted branch.

Device Authentication

The authentication of the device is fundamental in order to fight the presence of untrusted devices. The user communicating with a target device on a testing infrastructure, needs to be sure that the target is an authentic device, not a fake one coming from a counterfeiting process. Some countermeasures of this kind have been proposed in the literature.

In [4] , the authors propose a technique for device authentication. The user sends a challenge to the device. The challenge is sent to the key port of a stream cipher. The device computes the response using the initialization phase of the stream cipher. The user checks the associated response using a reference database. This way, the user is able to check if the

[4]: Rosenfeld et al. (2010), 'Attacks and Defenses for JTAG'

device has given the right response. The relation between the challenge and the response depends on the initialization value of the stream cipher, which is hardwired in the device using fuses. This configuration is secret and it is set at manufacturing time.

[58]: Das et al. (2013), 'Secure JTAG Implementation Using Schnorr Protocol'

The solution proposed in [58] , based on the Schnorr protocol, can be also used for the authentication of the device. This protocol has a bidirectional property that allows the authentication of both the user and the device. The already mentioned user authentication procedure can be performed on the other way around to allow the user to verify the authenticity of the device.

[69]: Dworak et al. (2014), 'Board security enhancement using new locking SIB-based architectures'

J. Dworak et al. proposed in [69] a technique to provide the authentication of a device mounted on a board. Each device owns a unique and secret ID number. When the tester wants to start a communication with a target device, the ID number is requested and checked against the correct one. An attacker, who wants to fake the target device, has to know the right ID associated with it. Hence, the ID must be kept confidential. For this reason, the ID number is encrypted in the transmission, in order to avoid other entities sniffing the JTAG network to steal its value. The encryption is performed by the device. The ID is added to the secret key, which must be as long as the ID. At the beginning of the authentication session, the user sends the secret key. In order to protect the key from sniffing, the sent key is obfuscated spreading it inside a random stream of bits. The obfuscation rule is secret and chosen at design time. A hardware module implemented inside the device performs the de-obfuscation of the received key.

The solution presented in [4] does not show high implementation cost, because the stream cipher used for computing the response is also used for encrypting test data, as we will explain later in this chapter. In [58], the ECC cryptography needed for the implementation of the Schnorr protocol leads to use a high amount of resources. The solution presented in [69] proposes the obfuscation of the key, which does not provide high security. Moreover, the constraint of implementing the obfuscation algorithm inside the device at design time does not permit flexibility.

Data Integrity

When test data are sent to a target device in the test infrastructures, these are often shifted through other IP cores or device that are connected to the same network. Malicious devices can corrupt test data in order to falsify test procedures and tamper with the communication. Granting the integrity of the communication allows the user and/or the device to be sure that the exchanged data have not been modified during the transmission. One common technique consists in using a Message Authentication Code (MAC) appended at the end of each transmitted message. The MAC is a unique signature that is computed as a function of the message content. The most used MAC algorithm in this field is the Hash MAC (HMAC). The HMAC is based on hash functions, such as SHA-256. When a message is received, the device internally computes the HMAC signature of the message. If it is equal to the signature that has been received appended to the message, it means that the message is intact. In the opposite case, it means that the message has been tampered with. The security of this primitive lies in the shared secret key used by both the user and the device to compute the HMAC signature.

The countermeasure proposed in [4] also provides the integrity of the exchanged messages between user and device. This is performed by appending a MAC signature to the message. The key used to compute the signature comes from the internal stream cipher employed for data encryption and device authentication. A challenge is sent by the user, which is used as a key by the stream cipher. The initialization value is hardwired at post-production using fuses. The stream cipher produces 80 bits of keystream depending on the input key and the initialization value. The produced keystream is used as secret key for HMAC. The user knows the value of the key because he or she owns the challenge-response pairs that depend on the configuration of the fuses.

[4]: Rosenfeld et al. (2010), 'Attacks and Defenses for JTAG'

R. Elnaggar et al. proposed in [50] a countermeasure that provides integrity in IJTAG reconfigurable networks. Untrusted embedded instruments are supposed to tamper with data, when shifted through their TDR. Therefore, the authors of the present paper proposed to create an alternative path that circumvents untrusted instruments. The alternative path is inserted by the SoC integrator, which is supposed

[50]: Elnaggar et al. (2018), 'Securing IJTAG against data-integrity attacks'

being trusted. The implementation of this solution relies on doubling each untrusted TDR, adding a trusted dummy TDR in parallel. This way, when data are shifted through the untrusted TDR, the alternative path is activated and only the alternative trusted TDR is visible from the RSN.

The usage of the MAC for integrity is based on a shared secret between the user and the device. The MAC also provides a weak authentication of the user. In fact, an unauthorized user, who does not know the key to compute a valid MAC, can only send invalid messages to the device.

Attack Detection

All countermeasures presented so far aim to avoid attacks on the target system. However, some other techniques in the literature aim to detect the execution of attacks. This is achieved by on-chip monitoring of the user behavior. When the behavior of the user is considered illegitimate, the system detects an attack and it goes into protection mode.

Detection techniques can be divided in two categories. The first category, comprises all the detection methods based on *static* rules. As soon as these rules are not respected, the user is considered to be an attacker. The second category comprises methods based on *machine learning*.

Static Detection

Static detection techniques are based on rules that are set at design time. Static detectors are synthetized during the design flow of the device. These detectors usually take as input the patterns sent by the user. If the patterns are not considered compliant to a legitimate behavior, the user is classified as an attacker trying to exploit the test infrastructure.

[70]: Baranowski et al. (2014), 'Access Port Protection for Reconfigurable Scan Networks'

R. Baranowski et al. proposed in [70] a detection technique for filtering the access to the test infrastructure. This solution is based on sequence filters that are placed on the TAP controller. They prevent the access to protected instruments and restrain it for instruments that are not completely protected. The filters take as input the sequence of instructions and data at the TDI port to decide whether the access pattern is allowed

or forbidden. If the user tries to access a forbidden instrument, the operation is not allowed by the filter. The filters are deactivated by default to enable post-manufacturing test. After that, they can be activated by blowing fuses. Alternatively, an authentication mechanism could be integrated in order to manage the activation of the filters. Further developments of this technique have been presented in [71–75].

In [6] the authors proposed a detection technique for the identification of attempts to reverse engineer the IJTAG reconfigurable network. A checker counts the number of shift cycles that are performed by the user during the RSN configuration. A legitimate user knows the structure, hence the length, of the RSN. Therefore, the number of shift cycles necessary to configure the RSN are exactly known. On the opposite, when the attacker explores the RSN, he/she performs several attempts. Thus, an inexact number of shift cycles is necessarily spent in the configuration process. When the checker detects this situation, the user is considered an attacker.

[6]: Kan et al. (2016), 'Echeloned IJTAG data protection'

X. Ren at al. presented in [76] a detection technique based on representative sequences of instructions. These sequences are chosen at design time as representative of legitimate operations. If the behavior of the user goes sideways for long time with respect to the representative sequences, an attack is detected. In the implementation, a counter is associated with each representative sequence. When all counters stop incrementing, it means that a non-representative sequence is being performed by the user, thus the circuit is subject to an attack.

[76]: Ren et al. (2019), 'IC Protection Against JTAG-Based Attacks'

The countermeasure proposed in [50] can be expanded in order to also detect attempts of data tampering. Genuine instrument responses (not shifted through the internal TDR of the untrusted instruments) are compared with the responses coming from the internal TDR of the instrument. If a difference, is detected a tainted bit is set in an extra RSN. When the tester collects the test responses, the presence of a taint bit indicates that an instrument has tried to tamper with some data.

[50]: Elnaggar et al. (2018), 'Securing IJTAG against data-integrity attacks'

The rules that underlie the static detection techniques must be set at design time. This implies that it is not possible to change the access policies without a complete redesign of the detectors. In the solution [70] the possibility of performing

post-manufacturing test using different policies is contemplated. Nevertheless, once the filters are activated, it is not possible to obtain higher privileges on the test infrastructure anymore.

Machine Learning

Detection techniques based on machine learning require the implementation of on-chip binary classifiers. They are special circuits that are able to evaluate the sequences of instructions sent by the user and label them as *normal* or *abnormal* behavior. The classifiers must be trained before being operative. During the training phase, instruction sequences belonging to both categories are labeled and sent to the classifier. In this way, the classifier sets its internal classification parameters. After that, the classifier is able to successfully classify the sequences autonomously in the operative phase. Input data are pre-processed in order to obtain the so-called *feature vectors*, which are a different representation of the data. The feature vectors are the input of the classifier in both the training and operative phase.

[76]: Ren et al. (2019), 'IC Protection Against JTAG-Based Attacks'

In [76] , the authors presented two detection techniques based on machine learning. They proposed the deployment of two different classifiers, the *random forest* and the *Support Vector Machine* (SVM). The random forest classifier is based on decision trees. Each tree takes as input a feature vector (or a part of it) and outputs a binary value that corresponds to its classification. The result of each tree is then sent to a majority voter that establishes the final result. The feature vectors, given as input to the random forest classifier, are extracted by the executed JTAG instructions. These features are derived from static elements extracted by the instruction, plus one transition bit. The transition bit indicates if the transition from the previous instruction to the present one is typical or not. The SVM is a classifier that defines a decision boundary during the training phase. The decision boundary separates the two classes of samples such that the smallest distance between the decision boundary and any of the samples is maximized. The input of the SVM is a sequence of JTAG instructions. The optimal length of the sequences, which is equal to 4 instructions, has been determined empirically by the authors. The present technique has been further extended by the authors in [77–79].

While the detector based on the random forest classifier is able to provide a classification based on static features of the instruction under execution, the SVM relies on sequences of several instructions. This makes the classification based on the SVM more efficient against attacks that are unknown in the training phase. A common drawback of these two solutions is that each time a new attack is conceived, it could be necessary to perform the whole training process again. Moreover, machine-learning techniques show more efficiency if coupled with other protections. This is due to the fact that in some situations the detection can fail because the attack is not recognized. Once the classifier has detected that the user is performing an attack, the system must activate a locking feature or going into a protected mode.

Summary

In Table 3.1 on the next page, we have regrouped all the countermeasure categories and the known security threats from Chapter 2. The first conclusion that can be drawn is that there is no countermeasure able to cover all the existing threats. All user authentication and attack detection techniques are able to protect the test infrastructures against external threats, but they leave the internal threats uncovered. On the other hand, device authentication, data confidentiality and data integrity techniques are able to cover some kind of internal threats each. For the purpose of this thesis, it is important to notice that scan encryption and data integrity techniques are the only countermeasures that are able to thwart both external threats and sniffing (or tampering). For this reason, the contributions of this thesis moved towards scan encryption countermeasures, since they were considered promising techniques for achieving a complete protection for test infrastructures.

Table 3.1: Analysis of the protection granted from each countermeasure against the known threats

		External Threats		Internal Threats	
		TAP	**IJTAG**	**Sniffing**	**Tampering**
User Authentication		Yes	Yes	No	No
IJAG User Autentication		No	Yes	No	No
Privilege Based Authentication		Yes	Yes	No	No
Data Confidentiality	***Scan Encryption***	Yes	Yes	Yes	No
	Secure Configuration	No	No	Yes	Yes
Device Authetication		No	No	Yes	Yes
Data Integrity		Yes	Yes	No	Yes
Attack Detection		Yes	Yes	No	No

3.2 Scan Encryption

Scan encryption techniques have known a relatively recent development due to their promising characteristics. From a security perspective, they rely on data encryption based on symmetric ciphers. In this section, we introduce all the basic concepts that will be useful to the reader for deeply understanding the proposed scan encryption techniques. We start by providing a summary on *symmetric encryption* from a purely cryptographic point of view. After that, we show how the symmetric encryption concept can be applied to the *test flow* of integrated circuits. Finally, we present scan encryption techniques from the literature, which were the *state-of-the-art* before the contributions of this thesis came out.

Symmetric Encryption

Understanding some key differences between the existing ciphers that are used for symmetric encryption is of fundamental importance to appreciate the peculiarities of the proposed scan encryption techniques. We focus on symmetric ciphers since their operations (e.g., same key used for both encryption and decryption) fit with the requirements coming from the test communication scenario. Moreover, symmetric ciphers propose a lower cost in terms of area and computation time than asymmetric ciphers. First, we recap the rationale of symmetric data encryption. Then, we provide a brief introduction on block and stream ciphers in order to set the terminology and highlight the key features.

In general, a cipher allows the sender to transform an input message m, called *plaintext*, in a ciphered version c, called *ciphertext*, using a *secret key* k. The receiver must be able to rebuild m from c upon the knowledge of the same k. A cipher is composed of two functions: **E**, called *encryption function*, and **D**, called *decryption function*, such that:

- The encryption function takes as input the message m and the secret key k, and outputs a ciphertext c, so that $\mathbf{E}(k, m) = c$.
- The decryption function takes as input the ciphertext c and the secret key k, and outputs the plaintext m, so that $\mathbf{D}(k, c) = m$.

The encryption of a message followed by the decryption of the correspondent ciphertext must result in the initial message, i.e., $\mathbf{D}(k, \mathbf{E}(k, m)) = m$. Ciphers that are used for providing confidentiality in the test infrastructures are *stream ciphers* and *block ciphers*.

Stream Ciphers

Stream ciphers are based on a theoretical cipher, called *One Time Pad* (OTP). In the OTP, the secret key must be as long as the message m. The encryption function is defined as:

$$\mathbf{E}(k, m) = m \oplus k = c \tag{3.1}$$

the decryption function is defined as:

$$\mathbf{D}(k, c) = c \oplus k = m \tag{3.2}$$

If k is perfectly random (i.e., according to the uniform distribution), the OTP has perfect secrecy. This means that the produced ciphertext is indistinguishable from a random sequence (this is due to the properties of the `XOR` operator). In this case, it is impossible for an attacker that intercepts the ciphertext to derive any information neither on the message nor on the key. However, from a practical point of view, the OTP is not implementable because of the key length. In fact, it is impracticable for the sender and the receiver to exchange a secret key that is as long as the message they want to transmit. Stream ciphers are implementations of the OTP. Instead of processing a random key k that is as long as the plaintext, a Pseudo-Random Generator (PRG) generates a

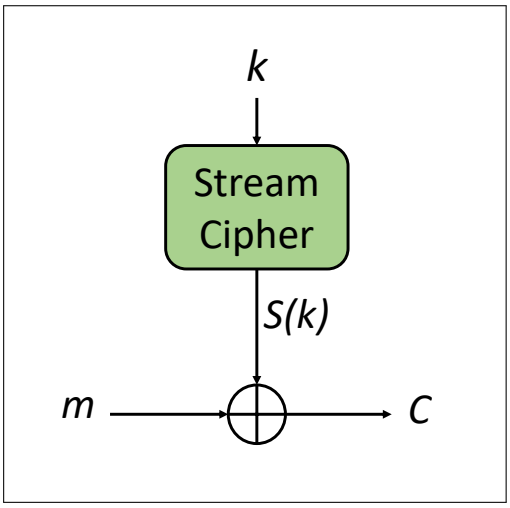

Figure 3.2: High-level architecture of a generic stream cipher

pseudo-random sequence of bits called *keystream*. The PRG takes as input a value k, called seed of the stream cipher, and outputs the keystream $S(k)$ (Figure 3.2). The encryption and decryption functions are thus defined as:

$$\begin{aligned} \mathbf{E}(k, m) &= m \oplus S(k) = c \\ \mathbf{D}(k, c) &= c \oplus S(k) = m \end{aligned} \tag{3.3}$$

As far as the PRG produces a keystream that is unpredictable, the resulting stream cipher is considered secure.

[80]: De Canniere et al. (2005), *TRIVIUM Specifications*

The TRIVIUM [80] stream cipher is widely used in the context of scan chain protection due to its low implementation cost (i.e., 3 AND gates, 11 XOR gates and 288 flip-flops). It is based on a *Non-Linear Feedback Shift Register* (NLFSR) used as PRG. The seed of the TRIVIUM PRG is made by an 80-bit secret key K, and an 80-bit *Initialization Value* (IV), which is publicly known. The generated keystream is denoted as $S(K, IV)$.

Block Ciphers

Block ciphers are based on mathematical objects called *Pseudo Random Permutations* (PRP). They are invertible functions that take as input an n-bit value m and a secret key k, and output an n-bit value c. A PRP is considered secure if, fixed a key k, the resulting function is indistinguishable from a random bijective function on n-bit values. Block ciphers implement a secure PRP. They are made of an encryption function that is able to encrypt a plaintext block into a ciphertext block using a secret key; and a decryption function that performs

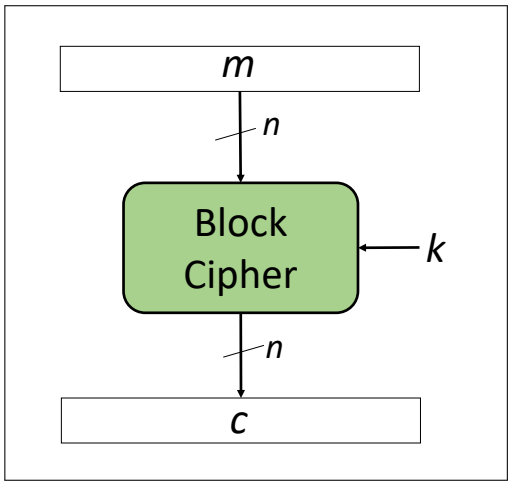

Figure 3.3: High-level architecture of a generic block cipher

the inverse operation and retrieves the plaintext block from the ciphertext (Figure 3.3).

One of the most common block ciphers is the Advanced Encryption Standard (AES), which we have already encountered in Chapter 2. However, while in Chapter 2 we have discussed how crypto-processors, such as AES, can be the victims of scan attacks, here we are using cryptographic circuits in order to *protect* the circuits against such attacks. Other block ciphers have been proposed, which target lightweight hardware implementations. For instance, PRESENT and SKINNY block ciphers have a lower cost in terms of area and power consumption.

Testing with Encrypted Data

Symmetric ciphers can be easily employed to set up a secure test flow. In fact, the test process can be seen as a communication between a tester and a device. The tester can be an authorized user accessing the test infrastructure in-the-field, or an ATE performing post-manufacturing test. The target device can be an integrated circuit, or a specific IP core inside a SoC that has its internal test infrastructure protected. The designer chooses a secret key that is stored inside the device. Subsequently, this key must be handed out to all the parties that are authorized to access the test interface of the device. Figure 3.4 shows a scan encryption scheme that could be employed for post-manufacturing test. It consists of the following steps:

1. the designer generates test patterns for the device and computes the expected *fault-free* test responses;

2. test patterns are encrypted by the designer resorting to the chosen cipher and using the secret key that has been stored inside the circuit under test;
3. the tester receives the encrypted test patterns that are shifted into the device during the test procedure. The encrypted test patterns are decrypted internally through the input scan cipher;
4. once the test has been applied, test responses are encrypted resorting to the output scan cipher;
5. encrypted test responses are decrypted off-chip to obtain the actual responses of the circuit and compare them with the expected ones.

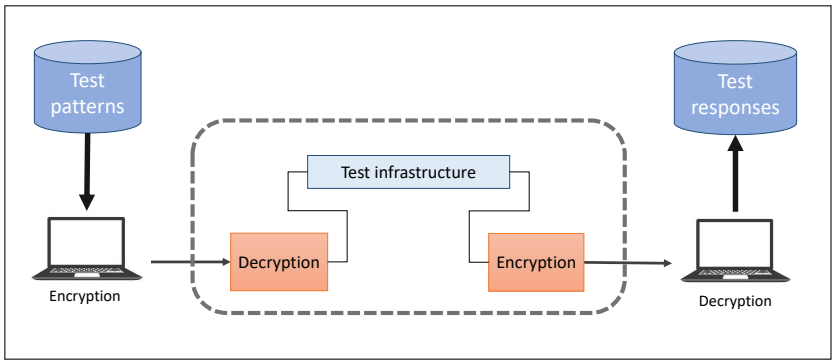

Figure 3.4: Functional scheme of scan encryption

The scan encryption technique merges both user authentication and data confidentiality into a unique security countermeasure. In fact, any malicious device or malicious user trying to sniff the test channel is not able to understand the content of the transmitted message. Moreover, an unauthorized user that does not know the secret key it is not able to successfully encrypt test data. It is worth pointing out that the input scan cipher is always there. Hence, its decryption function cannot be avoided along the test channel. For this reason, the only way to successfully communicate with the target device is to know the secret key and to properly encrypt all data that is introduced through the TDI pin.

Scan encryption schemes can be implemented using both block and stream ciphers. In the following, we will describe all the scan encryption techniques, based on both ciphers, that existed in the literature before the contributions of this thesis were published.

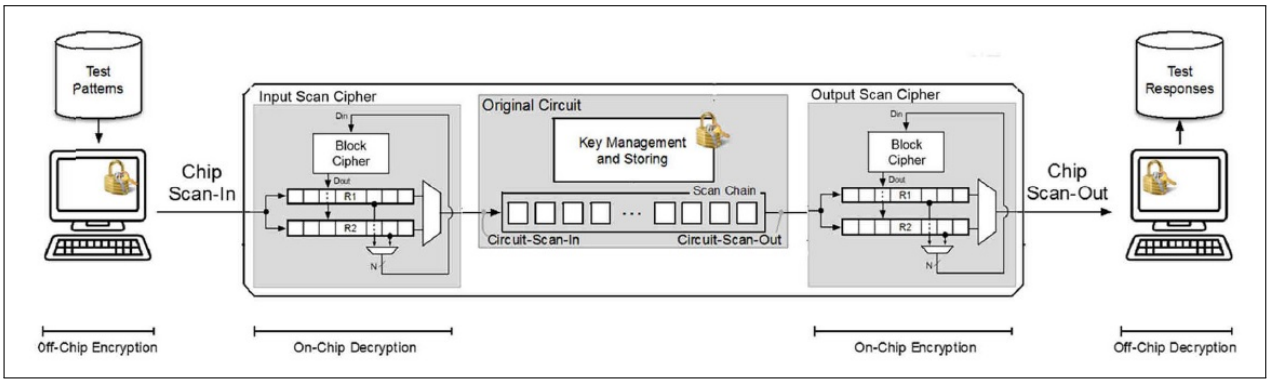

Figure 3.5: Scan encryption based on block ciphers [82]

Block Based Scan Encryption

Scan encryption based on block ciphers was proposed the first time in 2017 by Mathieu Da Silva et al. [81] . This work started from a collaboration between researchers from the Polytechnic University of Turin and our team at LIRMM laboratory. This first paper was well-acclaimed by the scientific community and, in 2019, it was extended with an important publication that extensively describes the scan encryption concept [82] . In addition, it shows how block ciphers can be employed to efficiently encrypt data that are involved in a test infrastructure based on single or multiple scan chains.

[81]: Da Silva et al. (2017), 'Scan chain encryption for the test, diagnosis and debug of secure circuits'

[82]: Da Silva et al. (2019), 'Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption'

As shown in Figure 3.5, two block-ciphers are added to the original circuit. The input scan cipher decrypts data sent by the tester, while the output scan cipher encrypts the test responses before transmission to the tester. Each cipher has two N-bit round registers ($R1$ and $R2$ in Figure 3.5) with two operating modes: `load` and `shift`. In `load` mode, the result of the round operation is stored. In `shift` mode, the round register is connected between the TDI/TDO pin and the internal test infrastructure. Two registers, alternating the `load` and `shift` modes, are essential in order to not incur in excessive test time overhead. This way, the scan ciphers can perform the decryption/encryption operations and the shifting operations in parallel. For instance, in the input scan cipher, while $R1$ is in `shift` mode and it receives new data shifted in from the TDI pin, $R2$ is used as round register of the block cipher and it stores the decrypted result of the test data block received during the N previous clock cycles. A controller is in charge of enabling the correct sequence of operations.

Test time takes $2 \times N$ extra clock cycles at the beginning of the

test procedure for loading $R1$ and $R2$ registers with the first two blocks to decrypt. The same additional overhead of $2 \times N$ extra clock cycles is required at the end of the test procedure to read-out the last response stored inside the $R1$ and $R2$ registers of the output scan cipher. The scan chain is filled one bit per clock cycle, thus the scan ciphers can be clocked at the same frequency as the scan chains. After each encryption, one segment consisting of N bits (i.e., one encryption block) is available on the round register and it is shifted into the scan chain in N clock cycles. Scan chains whose length is a multiple of N can be filled without penalties. However, it is also possible to manage the encrypted test with scan chains whose length is not a multiple of N, paying a penalty in terms of test time overhead.

Block cipher based scan encryption can be extended to multiple scan chains. However, this extension is not trivial, since it requires the implementation of a dedicated architectural solution. When multiple scan chains are present, each scan chain must be filled with a new bit in each clock cycle. Hence, the input block cipher must fill the scan chains one slice at a time. For this reason, the scan encryption operations must be performed at higher clock frequency. In fact, two clock sources are needed in the multiple scan chain scenario: a slower test clock for the scan chains and a faster scan encryption clock for the scan ciphers.

Block based scan encryption can be deployed after that the whole test infrastructure has already been designed. In fact, the scan encryption feature is a wrapper (made of the two block ciphers and the controller) that can be added to the circuit in the final steps of the design flow. Block based scan encryption can be easily applied to test compression methods as well, regardless of the test decompressor and compactor that are used. The tester generates the compressed stimuli used to test the device. These compressed stimuli are encrypted by the tester and decrypted by the device before being decompressed. The decrypted test stimuli are then processed by the test decompressor. The encryption of the scan output is performed on the already compacted test responses. Therefore, it is possible to conclude that block based encryption is transparent to any kind of advanced test infrastructure.

Stream Based Scan Encryption

Scan encryption techniques based on stream ciphers are all implemented with the TRIVIUM cipher. The TRIVIUM stream cipher is based on a NLFSR that takes a seed as input and produces a pseudo-random keystream as output. The seed is made by an 80-bit secret key and an 80-bit Initialization Vector (IV). While the key is secret, the IV can be public. The only requirement is that the same IV should not be used more than once.

K. Rosenfeld and R. Karri proposed in [4] an encryption technique that targets the JTAG infrastructure. The IV of the TRIVIUM cipher is hardwired on the device using fuses. The configuration of the fuses is secret and established at manufacturing time and it never changes during the device lifetime. The key is established before each encryption session with a challenge-response protocol. The user sends a challenge to the device, which is sent to the key input of the TRIVIUM cipher. This is run for a specific number of clock cycles in order to generate the first 80 bits of the keystream. This 80-bit value is used as the encryption key for the communication with the user. Unauthorized users that do not know the hardwired IV are not able to derive the encryption key used for the encryption.

[4]: Rosenfeld et al. (2010), 'Attacks and Defenses for JTAG'

In [5] , the same approach is used for securing the test in IEEE 1500 compliant IP cores. The encryption is performed with the TRIVIUM stream cipher. The management of the IV is not specified by the authors. The secret key is chosen randomly by the user and loaded into the cipher through a dedicated scan chain designed in order to prevent other IP cores from sniffing the key. In this case, the scan encryption technique does not provide user authentication, but it only protects test data against sniffing from malicious devices.

[5]: Rosenfeld et al. (2011), 'Security-aware SoC test access mechanisms'

S. Kan et al. proposed in [6] the encryption of the IJTAG reconfigurable network. Also in this case, the TRIVIUM stream cipher is used. The secret key and the IV are both fixed and stored inside the device.

[6]: Kan et al. (2016), 'Echeloned IJTAG data protection'

The presented stream based scan encryption techniques are chronologically older than block based solutions. In fact, stream ciphers have been preferred by the researchers due to their lower cost and their easiness of implementation. However, in the following of this thesis, we show that the

choice between block and stream ciphers is not as easy as it looks like.

Summary

In this chapter, we have identified the scan encryption as a promising technique for securing test infrastructures. In fact, scan encryption is able to protect test infrastructures from external attacks and internal attacks based on the sniffing of test data. Furthermore, the usage of symmetric encryption primitives makes it a low-cost technique with respect to complex authentication mechanisms based on asymmetric cryptography. We have identified two different approaches for scan encryption. One is based on block cipher encryption, the other is based on stream cipher encryption. Block based scan encryption has been the subject of some publications that thoroughly analyzed its implementation costs and its security properties. On the other hand, stream based scan encryption has been proposed by different authors, but its security properties have not been the object of extensive research. From an implementation point of view, stream ciphers have a smaller area cost with respect to block ciphers. Furthermore, the serial interface of stream ciphers better fits with the data flow in test infrastructures. For these reasons, we believe that stream based scan encryption should be better evaluated and considered for scan encryption implementations.

Secure Stream Based Scan Encryption

4

Existing stream cipher based scan encryption techniques (all based on the TRIVIUM algorithm) present a lower cost solution than scan encryption techniques based on block ciphers. In Chapter 3, we have shown that block based solutions must be accompanied by additional circuitry in order to cope with the serial interface of the test infrastructures. This overhead is particularly evident when dealing with multiple scan chains. Stream based solutions do not have this problem, but the existing implementations have not been thoroughly developed like it was done for block based solutions. For this reason, our research moved towards an exploration of new opportunities for employing stream ciphers in the scan encryption but, to our surprise, some problems were found out.

In this chapter, we show that the choice between block and stream based scan encryption is not as easy as it seems. At first, we show that the existing stream based techniques are subject to a serious vulnerability which makes them completely unsecure. After that, we propose new stream based scan encryption techniques targeting different kinds of test infrastructures and supporting a test protocol that can be employed at any level of the design hierarchy. Finally, we compare the implementation costs of both block and stream based scan encryption, and we will show that the best choice is not straightforward [83–86] .

[83]: Da Silva et al. (2018), 'A New Secure Stream Cipher for Scan Chain Encryption'
[84]: Valea et al. (2019), 'Encryption-Based Secure JTAG'
[85]: Valea et al. (2019), 'Stream vs block ciphers for scan encryption'
[86]: Merandat et al. (2019), 'A Comprehensive Approach to a Trusted Test Infrastructure'

4.1 Scan Encryption Vulnerability

The stream cipher is considered secure as far as two conditions are satisfied:

1. the keystream is unpredictable by the attacker;
2. the same keystream must not be used more than once.

If one of these conditions is not respected, it is possible to perform an attack, called *two times pad*. Let us suppose that two messages t_1 and t_2 are encrypted using the same

keystream $S(k, IV)$. If the attacker is able to observe the two resulting ciphertexts, the following operation is possible:

$$(t_1 \oplus S(k, IV)) \oplus (t_2 \oplus S(k, IV)) = t_1 \oplus t_2 \tag{4.1}$$

This means that simply performing an XOR operation between two ciphertexts results in the XOR of the two original plaintexts (i.e., their Hamming distance). This information can be exploited in many scenarios. For example, in the case of textual plaintexts, it is possible to retrieve some statistical information from the Hamming distance of the two messages that can lead to retrieve the content of the two plaintexts. However, in the context of test infrastructures, we already know a scenario where an attack of this kind would be detrimental for the security of the scan encryption. Let us consider the differential scan attack on the AES crypto-core. As we have detailed in Chapter 2, the attack is performed computing the Hamming distance of two consecutive partial responses. Let us assume that t_1 and t_2 are two partial encryption results that have been shifted out from the scan chains of the crypto-processor. It is evident that the two times pad attack allows the attacker to obtain the desired Hamming distance information, even if the single test responses are encrypted. Assuming that the stream cipher can be reset between two test sessions, the attack will be carried out as follows:

1. a plaintext m_1 is sent as input to the device;
2. after the first round of computation, the result of the first encryption round is stored into the round register, and the circuit is switched to test mode;
3. the content of the round register is shifted out of the scan chain and is encrypted by the stream cipher before being delivered to the circuit output;
4. the circuit is reset, in order to force the stream cipher to generate the same keystream again;
5. the same procedure from points 1 to 3 is performed using a second plaintext m_2, related to m_1;
6. the XOR operation between the two encrypted test responses is performed.

The obtained result is equivalent to the Hamming distance of the two unencrypted test responses. This means that the scan encryption countermeasure, if implemented in this way, is totally useless for protecting the circuit from the differential

scan attack.

The three implementations of stream based scan encryption, presented in Chapter 3 for securing scan chains, can all be exploited to perform the two times pad attack. In the proposal from K. Rosenfeld and R. Karri [4] , the key of the TRIVIUM cipher is established by the user through a challenge-response protocol. The IV is fixed. This means that if the user sends the same challenge to the device twice, he/she is sure that the same key is used twice for encrypting test data. Thus, the attacker can force the device to generate the same keystream, even without knowing its value. The technique described in [5] allows the user to directly set the stream cipher key. In fact, this solution does not guarantee user authentication. In this case, we can only suppose that the scan attack could be performed by a sniffing device. The two times pad vulnerability is present if the user does not change the key before sending a new message to the device. The countermeasure presented in [6] is based on a stream cipher whose secret key and IV are either hardwired with fuses or produced through a challenge-response procedure based on PUFs. Each device has a unique set of secret keys and IVs assigned to the different instruments of the IJTAG reconfigurable network. Unfortunately, the authors do not specify any precaution taken to change values of the keys or the IV between different encryption sessions. Hence, the two times pad attack can be carried out, destroying the effect of the scan encryption scheme.

[4]: Rosenfeld et al. (2010), 'Attacks and Defenses for JTAG'

[5]: Rosenfeld et al. (2011), 'Security-aware SoC test access mechanisms'

[6]: Kan et al. (2016), 'Echeloned IJTAG data protection'

It is clear that state-of-the-art implementations of stream based scan encryption have underestimated the care for the right key management that is demanded by stream ciphers. On the other side, block based scan encryption does not suffer such a weakness. In fact, block ciphers can reuse the same key all along the lifetime of the device without incurring into known security problems. This is a first indication of the fact that preferring stream based scan encryption is not self-evident as it seems at first glance. However, we have decided to give another chance to stream ciphers and we have proposed a new way of implementing stream based scan encryption.

4.2 New Secure Scan Encryption Solution

In order to overcome the vulnerability showed in the last section, we proposed a new scan encryption solution based on stream ciphers. We assume that the original circuit embeds at least one cryptographic IP core, a secure storage for all the secret keys, and a Secret Key Management Unit (SKMU). We also assume that the circuit implements a scan chain and at least some FFs of the cryptographic core are included in the scan network.

The proposed countermeasure is shown in Figure 4.1. It consists in adding two stream ciphers at the input and at the output of the scan chain respectively. An attacker unaware of the secret key used for encryption of the test data is not able to load the patterns he/she wants inside the scan chain. Moreover, the attacker is not able to understand the content of test data that are shifted out from the test infrastructure. Only users with the knowledge of the secret key are authorized to access the scan chains. The secret key of the stream ciphers is stored and managed by the SKMU of the protected crypto-core. By re-using the key management of the original circuit, the solution does not introduce new issues in handling the secret key. The SKMU delivers a dedicated scan encryption key to authorized users. The IV used to initialize the stream cipher is generated by a TRNG. This random IV is sent to the external user through the TDO pin. This way, the user is able to properly encrypt test patterns. The IV is totally random and it is different after each circuit reset, but it is not secret. The only secret is the key, which is known only by the authorized users. Since the stream ciphers are initialized with a different IV at each device initialization, the same keystream is never generated twice. Thus, this solution is not vulnerable against the two times pad attack, preventing any attacker from carrying out differential scan attacks. The first step of the new test procedure consists in generating test vectors for the DUT, and computing the expected test responses by simulation. Before any scan operation, the stream ciphers are initialized by generating a random number used as IV. The tester retrieves the generated random IV through the scan-out pin, and he/she encrypts test vectors off-chip using the random IV and the secret key. Once the stream cipher initialization is finished, the tester can shift the encrypted

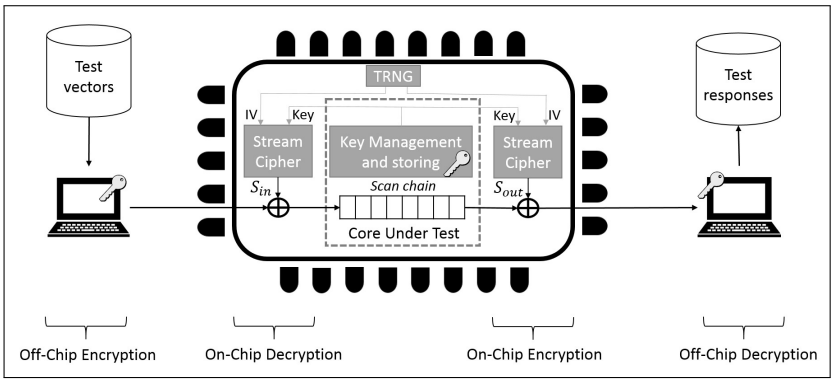

Figure 4.1: Schematics of the proposed stream based scan encryption. A TRNG produces the IV that is used to seed the stream cipher, together with the secret key.

test vectors into the test infrastructure. Each encrypted test vector is first decrypted using the keystream S_{in} generated by the stream cipher. After that, it is shifted into the scan chains of the device. Test vectors are applied to the circuit under test and test responses are obtained. During the shift out operation, the test responses are encrypted with the keystream S_{out} generated by the stream cipher placed at scan output. The encrypted test responses are shifted out of the circuit in order to be decrypted off-chip by the tester. Once decrypted, the test responses can be compared with the expected ones.

Nevertheless, the proposed test procedure shows inefficiency when multiple devices must be tested in parallel. This is the case when in-wafer test is performed during manufacturing. Newly fabricated integrated circuits are tested the first time when the silicon dies are still on the wafer. In this scenario, several dies are tested in parallel by applying the same patterns using flying probes. As explained above, implementing the proposed scan encryption technique forces the applied test patterns to be unique for every single circuit. In fact, they must be encrypted resorting to a random number that differs from one circuit to the other. Therefore, the proposed solution cannot be used for parallel testing of multiple circuits. To thwart this disadvantage, we propose to disable the use of the TRNG during the manufacturing process and to use a predefined hardwired IV for all circuits. This way, all keystreams are identical and all test patterns can be encrypted in the same way. Thus, we need a system that is able to electrically bypass the TRNG during in-wafer test. We propose to use *in-wafer sensors*, which are able to automatically determine whether the die is still attached to the wafer. These sensors are either based on *one time programmable memories*, or on the so-called *saw bow*. The latter

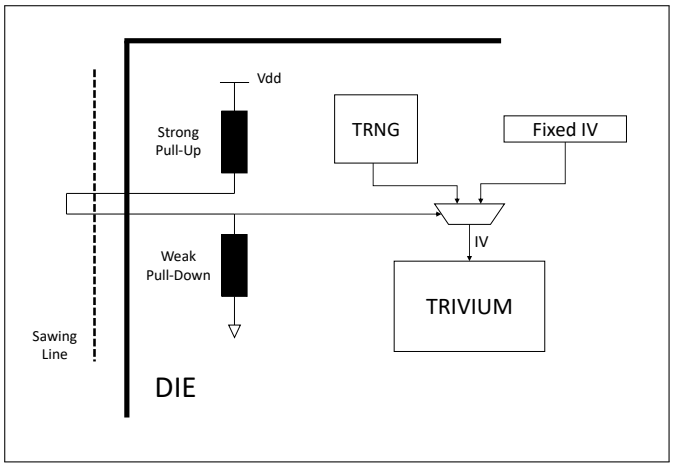

Figure 4.2: Schematics of the saw bow technique for isolating the TRNG when the die is still connected to the wafer.

is based on an electrical connection made by two resistive elements: a strong *pull-up* and a weak *pull-down* (see Figure 4.2). These elements are physically interconnected by a metal line across the sawing lines of the wafer. The strong pull-up resistance sets a logic value on the line when the sawing line is intact. When the dies are sawed, the weak pull-down resistance sets the opposite value on the line [87] . This way, a fixed IV can be used during in-wafer test. However, as soon as the dies are cut off from the wafer, the TRNG becomes the only IV generator on the IC.

[87]: Di Natale et al. (2017), 'Manufacturing Testing and Security Countermeasures'

In the following, we present two different implementations of this new stream based scan encryption solution. The first implementation targets very simple test infrastructures, where scan chains are directly interfaced with the external world. The second implementation has been developed to propose a modified TAP interface for JTAG infrastructures. In this case, we show a new JTAG implementation that is able to efficiently support the proposed stream based scan encryption solution. Both implementations are accompanied by experimental results.

Secure Scan Chain

We propose to encrypt the scan chain content using the TRIVIUM stream cipher. The keystream is generated relying on an 80-bit secret key and an 80-bits IV. This stream cipher has been chosen due to its low cost implementation. Moreover, the TRIVIUM cipher allows the designer to implement the generation of different keystreams using the same cipher (see Figure 4.3). This way, there is no need of implementing one

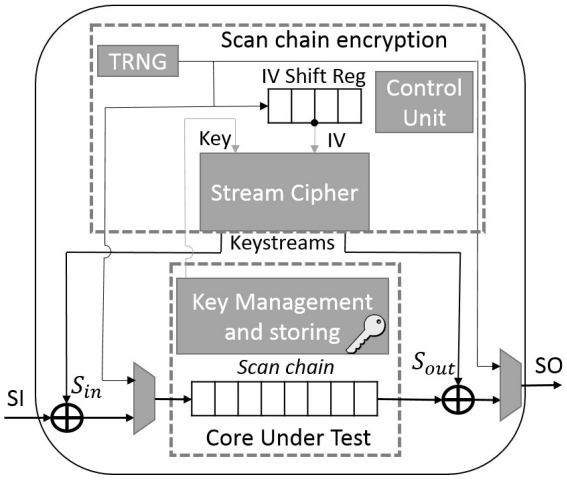

Figure 4.3: Architecture of the stream based encryption based of a scan chain infrastructure.

stream cipher at scan-in and another at scan-out. In fact, the implementation of one TRIVIUM is sufficient to generate two non-correlated keystreams for a marginal additional cost of 3 `AND` gates and 11 `XOR` gates [80] .

[80]: De Canniere et al. (2005), *TRIVIUM Specifications*

One TRIVIUM stream cipher generates the keystream S_{in} for the decryption of test vectors at scan input, and the keystream S_{out} for the encryption of test responses at scan output. An initialization procedure must be run each time the device is powered-on. This implies the presence of a test time overhead at the beginning of the test procedure. The TRNG must have a sufficiently high entropy before starting the generation of really random bits. This implies a first term in the test time overhead $T_{TRNGinit}$. Once the randomness of the generated bits has been reached by the TRNG, the IV is shifted into a dedicated shift register (`IV Shift Register` in Figure 4.3). At the same time, this is shifted out from the device by the tester. The second term in the test time overhead, called $T_{IVshifting}$, is due to the time needed to shift the IV into the `IV Shift Register`. Finally, the last term in the test time overhead, called $T_{SCsetup}$, is due to the initialization of the TRIVIUM stream cipher with the generated IV and the secret key, which is securely stored in the circuit. In the case of the TRIVIUM stream cipher, $T_{IVshifting}$ is equal to 80 clock cycles, since the IV is 80 bits long. $T_{SCsetup}$ is equal to 1152 clock cycles. This is the time needed by the TRIVIUM stream cipher in order to correctly set the initial state of its internal NLFSR. Once the initialization process is completed, the stream cipher is ready to generate the two keystreams for test data encryption and decryption. During the following phases of the test procedure, the proposed solution does not infer any additional overhead

on the test time. In fact, the encryption and decryption operations simply consists in combinational circuits (`XOR` gates) interposed between the external and the scan chain, thus no timing penalty is inflicted. The proposed architecture for the stream based scan encryption is represented in Figure 4.3. This implementation is composed of a TRNG, a shift register containing the generated IV, the stream cipher and a control unit.

At circuit reset, the control unit starts the initialization of the scan encryption as soon as the circuit is switched from normal to test mode. During the entire initialization process, the scan chain is kept inaccessible since the stream cipher does not generate the keystreams. Both the scan-in and the scan-out pins are connected to the TRNG through the multiplexers that are visible in Figure 4.3. This way, an attacker is not able to insert malicious data inside the scan chain, nor to observe its content. The only thing that the attacker can see, is the bitstream generated by the TRNG and the IV value, when this is available. At first, the IV is stored into the `IV Shift Register` before the initialization of the stream cipher. Once the stream cipher setup is finished, the multiplexers are connected to the scan-in and scan-out pins. After that, the stream cipher generates the two keystreams. The keystream S_{in} is combined bit-to-bit to the test vectors at scan input. The keystream S_{out} is combined bit-to-bit to the test responses of the circuit. The control unit keeps the stream cipher encryption active only during test mode. If the circuit switches to normal mode, the scan encryption is frozen and resumed as soon as the circuit goes to test mode again.

The proposed countermeasure requires the implementation of a TRNG, a shift register, a stream cipher and a control unit. We have summarized the costs of these submodules in Table 4.1 on the facing page. We have expressed the cost in terms of footprint area (expressed in *gate equivalent*) and impact on the initialization time (expressed in clock cycles). The TRNG model has been taken from the DesignWare IP library from Synopsys [88], where they provide a TRNG model of 15000 GE. The other submodules have a total area cost of 2600 GE: 300 GE for the IV shift register, 2048 GE for the TRIVIUM stream cipher generating two keystreams and 252 GE for the control unit. Concerning the test time overhead, the TRNG initialization time is dependent on the implementation, thus

Submodules	**Area Cost (Gate Equivalent)**	**Initialization Time (clock cycles)**
TRNG	15 000	$T_{TRNGinit}$
IV Shift Register	300	$T_{IVshifting} = 80$
TRIVIUM	2 048	$T_{SCsetup} = 1152$
Control Unit	252	/
Total	**17 600**	$T_{TRNGinit} + 1232$

Table 4.1: Cost of the submodules composing the proposed countermeasure

Circuit	**Triple DES**	**Pipelined AES-128**	**Pipelined AES-256**	**RSA 1024**	**LEON 3**
Cell Area (μm^2)	187 494	367 926	669 193	464 415	1 902 095
Test Time (c.c.)	687 101	1 944 877	4 559 845	39 405 239	11 612 051

Table 4.2: Benchmark ICs used to evaluate the implementation cost of the TRIVIUM based scan encryption

it cannot be estimated at architectural level. The IV shifting and the TRIVIUM initialization take 1232 clock cycles in total. The most incumbent cost of the proposed stream based scan encryption solution is clearly the TRNG implementation. However, if the target IC already embeds a TRNG (this is a very likely scenario in ICs for secure applications), this can be re-used during test mode by the scan encryption architecture. In this scenario, the TRNG will not introduce any area overhead. For this reason, we will not consider the TRNG cost as an overhead brought by the scan encryption implementation. We have implemented the proposed stream based scan encryption technique on five different ICs that we use as a benchmark for the evaluation of security solutions in test infrastructures. In these target devices we can find four crypto-processors implementing both symmetric (triple-DES, pipelined AES with 128-bit and 256-bit secret key) and asymmetric (RSA with 1024-bit private key) cryptography, and one microprocessor based SoC for aerospace applications (LEON3). The synthesis has been performed using a 65 nm library on Synopsys Design Compiler [89]. Test patterns have been generated using the ATPG tool from Synopsys TetraMAX [90]. Table 4.2 shows the area and the test time of the chosen benchmarks in their unprotected version. The proposed stream based scan encryption solution consists of 532 combinational cells and 382 flip-flops, when synthesized on the same technology library, corresponding to a constant area cost of 5409 μm^2 to be summed to the total area of the target IC.

Table 4.3: Cost of the scan chain encryption with the TRIVIUM stream cipher on the chosen benchmarks

Circuit	Triple-DES	Pipelined AES-128	Pipelined AES-256	RSA 1024	LEON 3
Cell Area	2.88 %	1.47 %	0.81 %	1.15 %	0.28 %
Test Coverage	100 %	100 %	100 %	100 %	100 %
Test Time	0.18 %	0.06 %	0.03 %	0.003 %	0.01 %

Table 4.3 reports the overhead, in percentage, introduced by the stream based scan encryption. In the Triple-DES (i.e., the smallest circuit) the area cost represents an overhead of 2.88%. In the LEON3 processor (i.e., the largest circuit) the area of the proposed solution represents merely a 0.28% overhead on the total surface. The test time overhead introduced by the stream based scan encryption is due to the initialization time at the beginning of the test procedure. Without considering the TRNG initialization, the proposed solution requires 1232 clock cycles for the initialization. This extra test time represents an overhead of 0.18% on the smallest benchmark (i.e., the fastes to be tested).

We have used the ATPG for generating test patterns for the target ICs. The generated test patterns are able to achieve 100% of stuck-at fault coverage. The additional scan encryption logic must be tested as well. Obviously, scan chains cannot be inserted into the scan encryption logic, otherwise the internal states of the stream cipher could be observed by an attacker with access to the scan chains. Therefore, the test of the scan encryption logic must be performed differently. Luckily, the stream cipher can be functionally tested using the test patterns of the original circuit. In fact, since the TRIVIUM stream cipher is based on an NLFSR, a potential fault on the cipher logic is easily propagated to the generated keystreams. Therefore, during the test data encryption and decryption, the stream based scan encryption solution is tested simultaneously with the original circuit. We have validated this assertion on the benchmark circuits. The test sequence of each circuit has been applied to the stream based scan encryption. The test patterns are processed by the keystream at scan-in and the test responses are processed by the keystream at scan-out. As expected, the same test sequence detects 100% of the stuck-at faults in both the original circuit and the additional scan encryption circuitry. No additional test patterns are needed to test the proposed countermeasure. These results are marked in the Table 4.3.

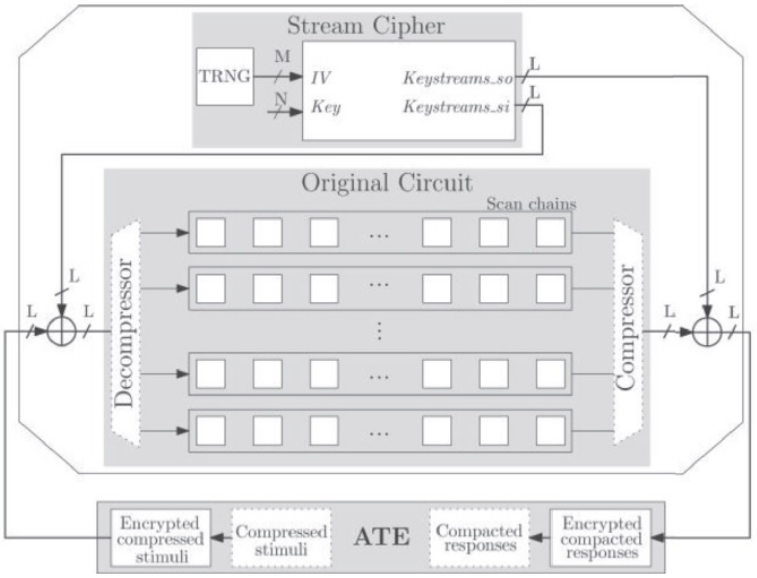

Figure 4.4: Stream based scan encryption applied to multiple scan chains supporting test compaction.

Multiple Scan Chains Implementation

The stream based scan encryption can be easily adapted to multiple scan chains. The only limitation is related to the number of keystreams that the stream cipher is able to generate. Let us consider a circuit where L scan chains are accessible through L scan-in pins and L scan-out pins, as shown in Figure 4.4. In this scenario, the stream cipher must generate two L-bit keystreams. The proposed solution can also be applied when a test decompressor is implemented at scan-in, and a test compressor at scan-out. In this scenario, the tester generates compressed test vectors for the circuit under test. Generated test vectors are encrypted with the TRIVIUM stream cipher. The encrypted compressed stimuli are shifted into the circuit and decrypted by the keystreams generated for the scan-in. The decrypted test stimuli are then applied to the test decompressor. The test responses are compressed before being encrypted on-chip with the keystreams generated for the scan-out. Finally, the tester decrypts the compacted test responses in order to compare them with the expected ones. The number of possible keystreams depends on the used stream cipher. For instance, the TRIVIUM can compute up to 64 keystream bits in one clock cycle. Therefore, 32 parallel test data can be decrypted at scan-in and 32 parallel test data encrypted at scan-out.

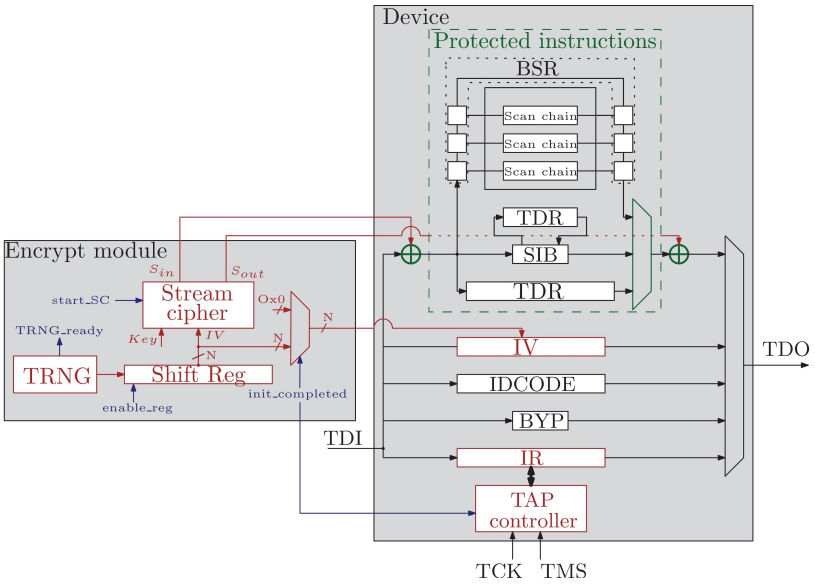

Figure 4.5: High-level architecture of Secure JTAG

Secure JTAG

The proposed stream based scan encryption is based on a new test protocol, which implies the circuit sending the IV to the user before starting the encryption. This implies a modification of the test protocol. When it comes to protecting simple scan chains, this problem can be easily solved adding a multiplexer and some additional control logic, as we have shown in the previous section. On the other hand, if we want to protect a standard test infrastructure based on the TAP interface, this becomes more challenging. For this reason, we have proposed a new secure JTAG infrastructure supporting the stream based scan encryption.

In Figure 4.5, it is possible to observe the high-level architecture of the secure JTAG infrastructure. This solution includes a TAP controller that gives access to the other test features included in the IC, such as scan chains (through the INTEST instructions) or IJTAG RNS. The main idea is to protect only a set of the JTAG instructions that are considered as *private* instructions. All the other instructions (i.e., *public* instructions) can be accessed by any user. The utilization of the proposed countermeasure consists in an initialization phase and an encryption phase. During the initialization phase, the TRNG generates the IV and sends it to the stream cipher circuit, in order to perform its setup. During this phase, the TAP controller locks the use of the private instructions and remains set on bypass mode. When the initialization phase is completed, the user can ask for the access to the private

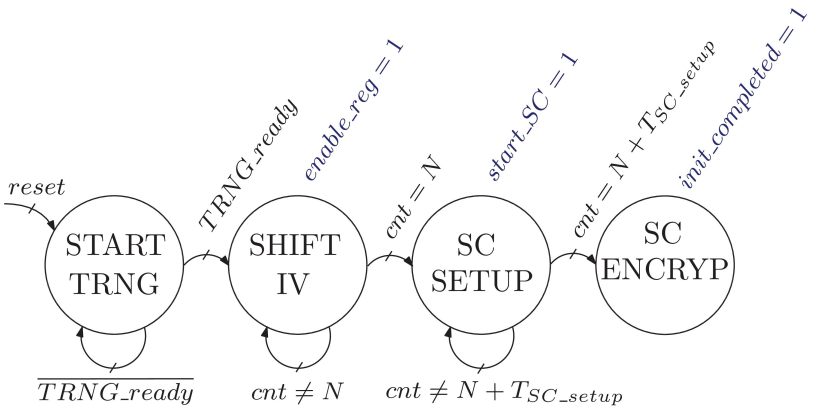

Figure 4.6: Finite State Machine controlling the initialization procedure

instructions. To do so, the user has to execute a specific instruction, called GETIV. When executed, this instruction connects a special register, containing the generated IV value, to the TDI and TDO pins. This way, the tester can shift out the IV that has been produced by the TRNG during the initialization phase. If the GETIV instruction is executed before the initialization phase is completed, a sequence of 0 values is returned as response. During the encryption phase, the tester knows both the secret key (if the tester is authorized) and the IV obtained via the GETIV instruction. At this point, it is possible to encrypt off-chip the test patterns using the IV recovered from the device. The test patterns, shifted through the TDI pin, are decrypted on-chip before being introduced into the corresponding TDR. During the scan-out operation, the test responses are encrypted on-chip. The tester collects encrypted responses from the TDO interface that can be decrypted off-chip, using the same IV and secret key used for the off-chip encryption. The solution can be extended to a whole set of protected instructions whose involved data need confidentiality. The designer has to define a set of private instructions. For example, the protected instructions can include (1) the INTEST instruction for accessing the scan chains; (2) the instruction accessing the IJTAG RSN, which includes critical instruments; (3) any instruction accessing a TDR containing confidential data, such as firmware updates of the device.

The stream cipher initialization procedure is controlled by an FSM, whose state transition graph is given in Figure 4.6. The

FSM is composed of four states (i.e., START_TRNG, SHIFT_IV, SC_SETUP, SC_ENCRYPT) and it outputs three control signals (i.e., enable_reg, start_SC, init_completed). All of them are initialized to the logic value '0'. At reset, the TRNG starts the initialization, while in the START_TRNG state. TRNGs have usually an initial set-up time during which the generated numbers are not random enough. They require some time to reach sufficient entropy. Therefore, during this period, the generated value cannot be used. As soon as the TRNG reaches a good entropy (i.e., $TRNG\_ready = 1$), the IV generation begins. During the SHIFT_IV state, the shift register (Shift Reg in Figure 4.5) receives the random bitstream generated by the TRNG ($enable\_reg = 1$). A counter (cnt in Figure 4.5) is launched at the same time. When cnt reaches the value N, the TRNG stops generating the random bitstream. N is equal to the number of bits of the IV. When the N bits of the random IV are generated, the TRNG is no longer used and it becomes available to other applications, if needed. Otherwise, it can be turned off. Once the counter have reached the value N, the control unit goes to the SC_SETUP state and starts the stream cipher initialization (i.e., $start\_SC = 1$). The FSM remains in this state for a time equal to T_{SC_setup}, needed for the stream cipher setup. Once the counter reaches the value $N + T_{SC\_setup}$, the initialization process is completed (i.e., $init\_completed = 1$). The stream cipher encrypts the data passing through the TDI and TDO pins of the protected TDRs. The keystreams are generated only in the case in which the TAP controller is in the Shift-DR state and a protected TDR is selected by the instruction under execution. In the other cases, the encryption is not needed, thus the stream cipher is deactivated and it generates no keystream. During the SC_ENCRYPT state, the user can write into a protected TDR. This is done by executing at first the GETIV instruction to read the IV, in order to encrypt data using the shared key. After that, the user executes the wanted private instruction, in order to access the corresponding TDR. The user places the IC into the Shift-DR state, where the stream cipher generates the keystream. The user shifts in the encrypted data, which are decrypted before being sent to the TDR. After that, the user places the IC into the Exit-DR state, in which the stream cipher stops the keystream generation. At the end of the operations, the TDR contains the plaintext data. The initialization process cannot be interrupted. The control unit ensures the setup completion before any possible operation

on the private TDRs can be executed. If a circuit reset occurs, the control unit is reinitialized and the TRNG generates a new IV for the stream cipher.

We have chosen the TRIVIUM stream cipher for the implementation of the proposed solution, due to its low area overhead. The security level that this cipher guarantees is enough for the target threat model. However, the proposed solution can be implemented with any kind of stream cipher, without changing the presented test protocol. This allows the designer to be able to immediately support any new stream cipher that would be released in the future. We do not consider the cost of the TRNG in the experimental results. If a TRNG is already implemented in the original circuit, the proposed countermeasure can exploit this TRNG during the initialization process, implying no cost overhead for the random number generation. After the generation of the IV, the TRNG is no longer useful for the proposed solution, and it can be used by another module of the SoC. On the other hand, if no TRNG is available in the circuit, implementing this technique implies an additional area cost. This has been estimated as 15000 GE, as is the case of the TRNG that we have taken from the Synopsys DesignWare IP library [88]. To evaluate the area overhead, we have considered a simple JTAG wrapper implementing a TAP controller, the instruction register, the bypass register and the IDCODE register. The JTAG wrapper is modified to include the GETIV instruction and its associated register. Moreover, some modules are added in addition to the modified JTAG wrapper: the TRIVIUM stream cipher, the register containing the random IV and the control unit. Table 4.4 on the following page reports the area cost of the proposed solution compared to the original JTAG wrapper without any security countermeasure. The induced area overhead is 500%. Obviously, this solution is intended for large SoC designs, where such an overhead on the TAP controller represents a negligible part of the footprint area. For instance, we have implemented the proposed solution on a LEON3 SoC, which has an area of 1,902,095 μm^2, when synthesized on a 65 nm technology library. Adding the secure JTAG solution increases the total area of 7794 μm^2, which corresponds to a 0.41% overhead on the area cost of the whole IC.

Concerning the test time cost, the proposed solution introduces an overhead that is only due to the initialization

Table 4.4: Area cost of the proposed countermeasure compared to an unprotected JTAG wrapper

Modules	Unprotected JTAG (GE)	Secure JTAG (GE)
JTAG Wrapper	625	1147
TRIVIUM	/	2048
IV Register	/	300
Control Unit	/	252
Total	**625**	**3747**

process. This process takes 80 clock cycles to shift the random IV into the dedicated register, and 1152 clock cycles for the TRIVIUM initialization. After this setup, the user must recover the IV executing the GETIV instruction before starting the encrypted test communication with the device. This takes 80 clock cycles for shifting the content of the IV register out. In addition to the time required to prepare the random number generation, the secure JTAG implies a test time overhead of 1312 clock cycles at the beginning of the test procedure. This test time overhead must be compared with the time taken to run the whole test sequence. For instance, the generated test patterns for the LEON3 SoC run in 11,612,051 clock cycles. Thus, the secure JTAG infrastructure introduces a 0.01% overhead on the total test time, without considering the initialization time of the TRNG.

The stream cipher based encryption of the JTAG interface does not affect the testability of the original circuit. The test coverage of the whole circuit is not reduced. However, the logic of the proposed countermeasure must be tested without the help of scan chains, because that would expose the stream cipher to scan attacks. We propose to functionally test the stream cipher using the same test data that are generated for the target device. Stream ciphers based on NLFSR, such as the TRIVIUM stream cipher, are easily testable because all the states of the stream cipher are correlated to the keystream content. The consequence is that errors generated by possible faults on the stream cipher are easily propagated to the circuit output during the encryption. To validate this assumption, we have evaluated the test coverage on the TRIVIUM stream cipher using the test sequences of some benchmark ICs. At scan-in, test patterns are processed by the input keystream generated by the stream cipher, and the test responses are processed by the output keystream. We have performed experiments with the test sequences targeting several circuits (i.e., pipelined AES-256, triple-DES, pipelined AES-128, RSA

1024 and LEON3 SoC). In all cases, the fault coverage for stuck-at faults in the secure JTAG is 100%.

4.3 Stream vs Block based Scan Encryption

In the previous sections of this chapter, we have shown that stream based scan encryption solutions are an appealing choice for providing security to test infrastructures. This is true at the condition of properly managing the IV generation, in contrast to what had been previously proposed by other authors. However, random IV generation does not come for free. In the presented techniques, we have proposed the implementation of a TRNG to generate an IV that is always different. Since TRNGs are often already present inside integrated circuits, we do not necessarily account this element as a cost that is exclusively due to the scan encryption presence. At this point, our research has been guided by the desire of giving a clear picture to engineers interested in implementing a scan encryption countermeasure. In this section, we present multiple implementations of both block and stream based scan encryption, in order to highlight the differences in terms of design and implementation costs between these two techniques.

Block ciphers are stronger primitives than stream ciphers. Differently from stream ciphers, they can perform the encryption of many plaintexts using the same secret key, without penalizing their security. However, according to their *mode of operation*, the attacker may be able to identify similarities between the ciphertext blocks. In fact, when the encryption is performed with a constant key, a certain block of plaintext is always transformed into the same ciphertext. Thus, if two plaintext blocks are equal, this relation is kept in the corresponding blocks of ciphertext. This mode of operation is called *electronic codebook* (ECB) mode, in which data are divided into blocks, which are then encrypted independently using the same key. Several other modes of operation exist for block ciphers in order to prevent the kind of vulnerability offered by the ECB mode. In the *cipher block chaining* (CBC) mode, the encryption of a plaintext block depends on both the key and all the ciphertext blocks that have been processed up to that point. CBC mode does not allow the

attacker to identify repeated data blocks in the encrypted communication, since each encrypted block of data depends on the previous ciphertext blocks. Differential scan attacks are ineffective no matter which block cipher mode of operation is employed in the scan encryption implementation. However, the ECB mode gives the possibility for an attacker to identify bits of interest in the encrypted test data. Let us assume that data shifted out from the device include bits of the round register of a crypto-processor. Stimulated with two different plaintexts, the crypto-processor will produce two different round register values, while the other data stored in the scan chain will remain the same. After the encryption of the shifted out scan data in ECB mode, non-changing data were encrypted in the same way (i.e., resulting in the same ciphertext). On the other hand, round register data, which differ due to the application of two different plaintexts, result in two different encrypted blocks, revealing the bits of interest in the scan chain. Consequently, running two different plaintexts can allow the attacker to identify the encrypted segments in the scan chain containing at least one bit of the round register. In the case where such information could allow the attacker to carry out a new threat, CBC mode is a more secure implementation. Block ciphers can also be configured in order to operate like stream ciphers. Block ciphers in *counter mode* (CTR) are used as pseudo-random generators in order to produce the keystream one block at a time.

[82]: Da Silva et al. (2019), 'Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption'

We have implemented the secure JTAG using the block cipher based scan encryption presented in [82] for performing test data encryption using lightweight block ciphers. Two block ciphers are implemented, one for the decryption performed at scan-in, another for the encryption performed at scan-out. Each of these two ciphers have two round registers with two operating modes, `load` and `shift`. The `load` mode is used to perform the encryption and decryption operations, while the `shift` mode is used to acquire the data shifted through the scan chain. The two operating modes are performed in parallel in order not to waste test time. In this case, differently from [82], we have implemented the block ciphers in CBC mode in order to provide a more secure implementation. The use of block ciphers implies the padding of test data to reach the block size. The scan chain is filled segment by segment, each segment consisting of N bits, i.e., the block size. When the scan chain length is not a multiple of the

block size, the solution still works, but each test pattern is padded with extra bits. Thus, the tester has to complete the shift operations employing additional clock cycles, resulting in a test time overhead on each test pattern. Let us consider a circuit with a number of flip-flops equal to:

$$F = S \cdot N + R \tag{4.2}$$

where S is the number of N-bit segments, and $R = F \pmod N$. The test time overhead $T_{overhead}$ introduced by the block based scan encryption over K test patterns is equal to:

$$T_{overhead} = \begin{cases} 2 \cdot 2N & \text{if } R = 0 \\ 2 \cdot 2N + (N - R)(K + 1) & \text{if } R > 0 \end{cases} \tag{4.3}$$

The authors in [82], propose a solution in order to do not waste the additional clock cycles used to synchronize the scan chain encryption scheme with constant segment length N. These $N - R$ extra clock cycles are actually exploited for testability improvement without requiring any additional test time. Indeed, by adding $N - R$ dummy scan flip-flops to the original scan chain, these extra flip-flops can be used as observation points to the circuit logic. The observation points permit to improve the propagation of test responses to the scan flip-flops, without any impact on the test time. The goal of the observation point insertion is to reduce the test sequence length, i.e., to reduce the number K of test patterns guaranteeing the same fault coverage. We present the results of this optimization showing that the additional flip-flops, used as observation points, can compensate the additional shift operations required by the scan encryption.

We compare block based and stream based scan encryption solutions using some benchmarks: triple-DES, pipelined 128-bit AES, pipelined 256-bit AES, 1024-bit RSA and the LEON3 SoC. In these first experiments, all the circuits are equipped with a single scan chain. These benchmarks have been synthesized using a 65 nm technology library. We have implemented the stream based scan encryption with the TRIVIUM stream cipher and two block ciphers in CTR mode: PRESENT and AES. The block based scan encryption has been implemented with the PRESENT and the SKINNY block ciphers in CBC mode. Concerning the stream based scan encryption, we do not consider the cost of the TRNG

Table 4.5: Area overhead of different block and stream based scan encryption techniques.

	Triple-DES	Pipelined AES-128	Pipelined AES-256	RSA 1024	LEON 3
AES-128 CTR	25.66 %	13.08 %	7.19 %	10.27 %	2.53 %
PRESENT CTR	3.64 %	1.86 %	1.02 %	1.46 %	0.36 %
TRIVIUM	2.88 %	1.47 %	0.81 %	1.15 %	0.28 %
PRESENT CBC	5.82 %	2.97 %	1.63 %	2.33 %	0.57 %
SKINNY CBC	5.43 %	2.76 %	1.52 %	2.17 %	0.53 %

Table 4.6: Test time overhead of different block and stream based scan encryption techniques.

	Triple-DES	Pipelined AES-128	Pipelined AES-256	RSA 1024	LEON 3
AES-128 CTR	0.020 %	0.007 %	0.003 %	0.0004 %	0.001 %
PRESENT CTR	0.014 %	0.005 %	0.002 %	0.0002 %	0.0008 %
TRIVIUM	0.18 %	0.06 %	0.03 %	0.003 %	0.01 %
PRESENT CBC	0.31 %	0.81 %	0.006 %	0.33 %	0.004 %
SKINNY CBC	0.31 %	0.81 %	0.006 %	0.33 %	0.004 %

in the experimental results. In the case where a TRNG is already implemented in the original circuit, the proposed countermeasure can exploit this TRNG during the test mode, implying no overhead for the random number generation. If a TRNG has to be implemented, the related area cost is evaluated to be 15000 GE from the Synopsys DesignWare IP library [88]. This value is equivalent to 31200 μm^2 on the 65 nm technology library adopted for the experiments.

Area overheads are reported in Table 4.5 for both stream and block based solutions. It must be noted that the block based scan encryption requires the implementation of two ciphers, one for decrypting input test patterns, the other for encrypting test responses. The stream based scan encryption requires the implementation of only one cipher to deliver two keystreams. Furthermore, in our experiments, we have not considered the key management, because we suppose to use the key management policy adopted in the target device. Concerning test time, we consider test sequences for the detection of stuck-at faults, obtained through the

ATPG from Synopsys TetraMAX [90]. Table 4.6 on the facing page reports the test time overhead of the proposed scan encryption countermeasures. The test time overhead of the stream based scan encryption is due to the initialization phase: the TRNG initialization, the shifting of the random IV and the stream cipher setup (in the case of the TRIVIUM stream cipher). Without considering the TRNG initialization, we obtain 138, 95 and 1232 clock cycles of initialization times for AES-128 CTR, PRESENT-128 CTR, and TRIVIUM respectively. In any case, they represent a marginal overhead on the total test time needed to test the whole IC. In block based solutions, test patterns must be padded in such a way that their total length is a multiple of the block size, i.e., 64 bits in the case of PRESENT and SKINNY. This induces a test time overhead for each pattern. This overhead can be reduced adding test points. The DfT tool from Synopsys TetraMAX [90] is used for selecting observation points in the circuit logic and for generating new test patterns. We have constrained the tool to use only the $N - R$ extra flip-flops for testability improvement, i.e., we have forced the scan chain length to be a multiple of 64. Observation points drive extra logic cones that connect them to the added scan flip-flops, thus allowing their observability at test time. For instance, the triple-DES core has 8808 scan flip-flops that can be divided into 137 segments of 64 bits. This implies the adding of 24 scan flip-flops with the corresponding observation points. The new test time cost has an overhead of 0.038% with respect to the unprotected scenario. If we would not have added the observation points, we would have lost 24 clock cycle on each test vector, inducing a test time overhead of 0.31%. Therefore, we can state that using this optimization based on observation points can decrease the test time overhead up to one order of magnitude. It is clear that scan ciphers outperform block ciphers in scan encryption implementations. This is mainly due to the difficulty for block based solutions to be adapted to the serial interface offered by all test infrastructures.

The scan encryption enables testing without incurring in fault coverage reduction, since the original test patterns are applied as they are. However, the logic that is involved in the scan encryption circuitry must be tested without the help of scan chains that would expose the ciphers to scan attacks. We propose to functionally test the ciphers using the same test patterns used to test the target device. The test of block ciphers, such as PRESENT, SKINNY and

AES, is facilitated by the diffusion properties of the crypto-algorithms, as described in the literature [91–93] . Stream ciphers based on NLFSRs, such as the TRIVIUM cipher, are also easily testable since all the states of the stream cipher are propagated to the keystream. Therefore, both ciphers easily propagate the possible errors to the circuit outputs when an encryption is performed. To validate the assumption, we have evaluated the test coverage on all the proposed ciphers (i.e., TRIVIUM, PRESENT, SKINNY and AES) by applying the test sequence of the target devices.In fact, both test patterns and responses are processed by the scan ciphers at scan-in and scan-out respectively. We have performed the experiments using the test sequences that had been generated for the benchmark circuits (i.e., pipelined AES-256, triple-DES, pipelined AES-128, RSA 1024 and LEON3 SoC). In all cases, the fault coverage for stuck-at faults in the scan encryption circuitry is 100%. In other words, the ciphers are tested for free, with no additional test patterns required.

When executing the test procedure, the tester has to encrypt and decrypt test patterns and responses off-chip. The off-chip decryption is performed for comparing the obtained responses with the expected ones. However, this computation is not always necessary and it depends on the choice between stream and block based scan encryption. In fact, expected test responses can contain unknown values, called *X-values*, describing an unknown binary state. These unknown bits are ignored during the comparison between the obtained test responses and the reference ones. Due to the confusion and diffusion properties of block ciphers, the block based scan encryption spreads the unknown bits to the entire encrypted response. For this reason, the off-chip decryption for block based scan encryption is mandatory in order to perform the comparison with the plaintext responses. In the case of stream based scan encryption, the unknown bits remain at the same position, since the encryption operates bitwise. Therefore, the unknown bits can directly be ignored in the encrypted test responses, avoiding the need for the tester to decrypt every obtained test response from the circuit. Therefore, the comparison can be performed between the encrypted responses and the encrypted expected ones.

Integrating scan encryption techniques in an IC design simply consists in adding ciphers at the input and output of the scan chains. In the case of stream based scan encryption, it

[91]: Schubert et al. (2000), 'On Random Pattern Testability of Cryptographic VLSI Cores'
[92]: Natale et al. (2010), 'Self-Test Techniques for Crypto-Devices'
[93]: Doulcier et al. (2008), 'AES-Based BIST: Self-Test, Test Pattern Generation and Signature Analysis'

also requires some modifications of the JTAG test wrapper, but no modification on the internal logic of the target device is needed. Therefore, both solutions can be applied without modifying the original circuit design. In the case of SoCs, each IP core has its test interface connected in a daisy-chain fashion. This way, the internal DfT resources of each core are serially connected to the TAP interface of the SoC. When the scan encryption is implemented on one IP core, all the other IP cores included in the test infrastructure receive test data in an encrypted form, thus providing protection against internal threats. The stream cipher encryption operates in a stream-like fashion, thus no issue is present when implementing stream based scan encryption on a test daisy-chain connection. Contrarily, in the case of block based scan encryption, test data are padded to a multiple of the block size. Thus, the extra data used for padding is shifted through the other IP cores in the test daisy-chain, resulting in possible security issues. Therefore, the designer has to be aware of the potential problems when the block based scan encryption is implemented. A way to avoid this issue is to design scan chains in order to have length multiple of the block size (e.g., resorting to adding observation points), thus modifying the DfT flow of the circuit design.

Summary

Here, we summarize the conclusions that we have drawn from comparing stream and block based scan encryption techniques:

Security both stream and block based scan encryption provide security against external attacks and internal attacks based on test data sniffing. In the case of differential scan attacks, the proposed stream based solution is the only existing secure scan encryption solution of this kind, because it does not allow the two times pad attack. Both solutions need a secret key integrated inside the target device, whose management can benefit from the key management policy that is provided by the target device.

Testability both techniques guarantee the full testability of the target device, both at manufacturing and in the field. Manufacturing test is not impacted by any loss of fault coverage induced by the adding of the scan encryption

solution. Stream based scan encryption needs a method to bypass the random IV generation when in-wafer test is performed. Test in the field is granted, because the access to the test infrastructure is left open, even though only authorized users can successfully communicate with the test infrastructure.

Area Cost stream based scan encryption implemented with the TRIVIUM stream cipher has an area cost which is one half of the area cost induced by the block based scan encryption implemented with the SKINNY block cipher (i.e., the lightest block cipher that we have used in our experiments). Actually, it is possible to observe that a lightweight block cipher, such as the SKINNY cipher, has the same area footprint of the TRIVIUM cipher. However, in the stream based scan encryption, it is sufficient to implement only one cipher generating two keystreams, while in the case of block ciphers, it is mandatory to implement a cipher at scan-in and a cipher at scan-out. This results in the stream based scan encryption costing the half of the block based one. On the other hand, if we account the cost of the TRNG as a cost induced by the implementation of the stream based scan encryption, its cost increases drastically and it becomes up to three times higher than a block based solution. Therefore, it is clear that stream based scan encryption is convenient from an area cost perspective only if a TRNG is already available in the target device implementation.

Test Time Cost stream based scan encryption has a big advantage on a timing point of view. This is possible because the stream cipher encryption benefits from its natural fit with the serial interface of the standard test infrastructures. For this reason, the test time penalty induced by stream based solution is a constant initialization time that is negligible with respect to the total test time of the target device. Block based scan encryption solutions need to adapt the block based datapath of block ciphers to the serial interface of the test infrastructure. For this reason, a strong penalty on the test time is induced by the need of padding test vectors with extra bits. This overhead can be reduced by adding test points, thus modifying the test infrastructure of the target device.

Multiple Scan Chains when multiple scan chains are

present, stream based scan encryption can benefit from its capability of generating multiple keystreams from the same stream cipher. For this reason, it is possible to manage up to 32 scan chains implementing only one TRIVIUM stream cipher. In the case of block based scan encryption, the management of multiple scan chains is definitely more complicated. In fact, block ciphers must be driven by a higher clock frequency than the rest of the test infrastructure. This leads to a higher power consumption and a deeper modification of the IC design.

5 Scan Encryption Security

Scan encryption provides protection against most of the known attacks targeting test infrastructures. On one side, the encryption of test data hinders the feasibility of attacks based on sniffing the communication channel between the user and the target device. On the other side, external attacks are not feasible, because the attacker does not have control over the data that are sent to the test infrastructure, thus he/she cannot understand the results produced by the target device and read out through the test infrastructure. The only capability that is left to the attacker is the possibility of shifting data through the scan-in pin, which are internally decrypted, thus unintelligible to the unauthorized user. Most threat models that we have presented in Chapter 2 are based on the assumption that the attacker can insert specific patterns inside the test infrastructure. However, we have identified some scenarios where even inserting random bits inside the test infrastructure could pose a security threat if appropriate countermeasures are not implemented. This is the case in some microprocessor based systems, where protected operational modes are activated and deactivated changing the value of a single bit in a register. In this case, even inserting random values inside the internal scan chains could lead to an attack, thus making the scan encryption countermeasure ineffective. In Chapter 3, we have shown several user authentication mechanisms aiming at hindering the usage of the test infrastructure by malicious users. We have seen that these techniques are mostly based on the insertion of a secret key or password, and on challenge-response protocols. On our side, we have identified an opportunity offered by the encryption, with the aim of obtaining a form of user authentication that goes beyond the traditional stand-alone techniques. Specifically, it is possible to rely on the device passively decrypting input data using a secret key. Therefore, assuming that an attacker has no control over the decryption process, it is possible to impose format rules that the entered data must comply with in order to be accepted by the device. By doing so, the device is able to check, after the decryption, if the data has been sent by an authorized user owning the secret key, or by a malicious user.

In this chapter, we discuss the security strength and weaknesses of scan encryption. At first, we are going to provide a security analysis of the scan encryption techniques that have been discussed so far. We show the efficiency of this technique against the known attacks. After that, we present a new attack model that exploits the capability, offered by scan encryption, of inserting random bits inside the scan chains. We discuss the context of the attack and its implementation on a simple microprocessor. Finally, we present a technique based on the insertion of *parity bits* in convenient points of the test data, in order for the device to check whether the input data come from an authorized user.

5.1 Security analysis against different attacks from the state-of-the-art

Scan encryption security revolves around two key points: *data confidentiality* and *user authentication*. The confidentiality of transmitted data is a key feature of encryption itself. In fact, when the user and the device exchange encrypted data with a secret key, no one else is able to understand the information transmitted. This means that any attack attempt made by malicious entities within the system, and connected to the testing infrastructure, cannot be successful. Furthermore, all attacks based on observation of the internal states of the circuit are made impossible by the fact that everything that is shifted out of the test infrastructure is encrypted. User authentication is obtained thanks to the fact that the knowledge of the secret key is fundamental to be able to communicate effectively with the device. If the user does not know the key that the device uses for decryption, the user will not be able to predict the form the incoming data will take once they arrive within the test infrastructure. This makes it impossible to force a specific state on the internal registers of the circuit. In fact, for the attacker it is impossible to predict the decryption result without knowing the secret key. Moreover, the decryption operation can be seen as a random permutation of bits. For this reason, the only capability left to the unauthorized user is to enter random data into the test infrastructure. Below, we are going to discuss the feasibility of the attacks mentioned in Chapter 2 on a system where scan encryption is implemented.

Differential Scan Attacks require that the attacker can switch the circuit in test mode at the appropriate time and extract the contents of the scan chains. Scan encryption does not prevent this practice. Nevertheless, the content of the scan chains is encrypted during the extraction, so the attacker is not able to perform the necessary post-processing to obtain the key. In this context, we have shown in Chapter 4 that stream based scan encryption requires a careful implementation of the initialization vector management, otherwise differential scan attacks remain implementable despite encryption. As far as *test mode only attacks* are concerned, these require the attacker to completely operate in test mode and insert the crpyto-processor's inputs through the scan chains. Scan encryption places a strong barrier in front of this practice, because the attacker loses control of the values sent to the crypto-processor and it is not possible to impose the desired values that are needed to successfully perform the attack.

Reverse Engineering is performed by stimulating the circuit logic with appropriate inputs and recording the respective responses. The use of scan encryption completely invalidates the feasibility of such an attack. The encryption and decryption operations that are performed on the data do not allow to obtain any correlation between input and output data.

Debug Access to a protected circuit with scan encryption is only allowed using the correct key. Usually, debugging is done using support software that automatically generates low-level commands to communicate with the TAP controller. The attacker who does not provide the correct key to the debugging software will not be able to send valid commands to the circuit. If we imagine a scenario where encryption only involves the internal scan chains, such as the Secure JTAG, an attacker may be able to establish a correct communication with the debug interface, however the extracted data will be encrypted, so it will not be possible to extract any useful information from the internal memory. However, the lack of an input integrity check may allow the insertion of random data that could, for example, lead to the insertion of corrupted data in the memory in an unpredictable way.

IJTAG Access is extremely limited if scan encryption is im-

plemented. To attack an IJTAG network, the first step is to reverse engineer the RSN and understand the organization of the various SIBs in order to access the different TDRs. In this step, the attacker progressively inserts bits with value '1' into the network and observes how the length of the network varies, to identify the position of the SIBs. By implementing scan encryption, this reverse engineering process is severely compromised, as it is extremely difficult, if not impossible, for the attacker to find useful patterns to perform an attack.

Sniffing the communication channel is a threat that is effectively removed by the presence of scan encryption. In fact, the encryption itself has the primary purpose of making the data unintelligible during their transmission. Any attacker who tries to spy on the transmitted data will not be able to obtain any information without knowing the key.

Tampering test data has been considered a danger by multiple attack models proposed in the literature. For instance, corrupting the test input and the corresponding responses can lead to falsified test outcome. In this scenario, scan encryption is only partially able to prevent possible attacks. In fact, it is possible for the attacker to modify the value of test data without the target device realizing it. Even if it is not possible to foresee the result of the applied corruption, it is however possible to apply a random corruption to test data.

5.2 Security Threat against Scan Encryption

Scan encryption allows the attacker to enter random data into the test infrastructure. This is possible because all data that the attacker tries to insert through the scan-in pin, are unpredictably corrupted by the input cipher. Unfortunately, the scan encryption techniques do not offer in any way the possibility to prevent unauthorized access. We show in the following how the insertion of random data may pose a threat. This occurs in cases where, in order to carry out an attack, it is sufficient to force, through the scan chains, a limited number of flip-flops to a specific value. If the rest of

the attack can be carried out in functional mode, the presence of scan encryption is not enough to prevent the attacker from entering, after a certain number of attempts, the desired value inside the device. In this thesis, we hypothesized such a scenario, using a microprocessor equipped with Trusted Execution Environment (TEE) as a victim device. In fact, in this kind of microprocessors, we have a single bit containing the binary value which determines the activation of the TEE. In the following, we briefly describe such systems, and we show a case study that we have conceived to perform some experiments.

Trusted Execution Environment

Nowadays, many microprocessor architectures are equipped with Trusted Execution Environments, which permit the execution of both *secure* and *non-secure* applications on the same microprocessor core. The ARM TrustZone is an example of TEE [94, 95]. Two separate execution environments are present in the processor: the *secure mode* and the *normal mode*. The code executed in secure mode is not accessible by the code executed in normal mode. This way, the same microprocessor core can securely execute code both in the secure and non-secure environments in a time-sliced fashion. The switch between these two modes is performed by setting a bit in a specific configuration register of the CPU. In the ARM TrustZone, this bit is called NS-bit (Non-Secure bit), which is placed in the *Secure Configuration Register*. When the NS-bit has value '1', the microprocessor executes code in the non-secure environment, and vice versa. The NS-bit can be overwritten exclusively by special kernel routines. Both peripherals and memory addresses are marked as secure or non-secure relying on an additional address bit. Only processes from the secure environment can access secure resources.

From now on, we refer to a generic microprocessor architecture equipped with TEE, as depicted in Figure 5.1. A *Security Bit* (SB) is placed in the *Configuration Register* (CR). The SB drives the switching between secure and non-secure environments. When the processor is in normal mode (i.e., *SB=0*), the applications executed from the user-space memory cannot access the protected addresses, where secured resources are allocated. The term *secured resources* can refer to both memory

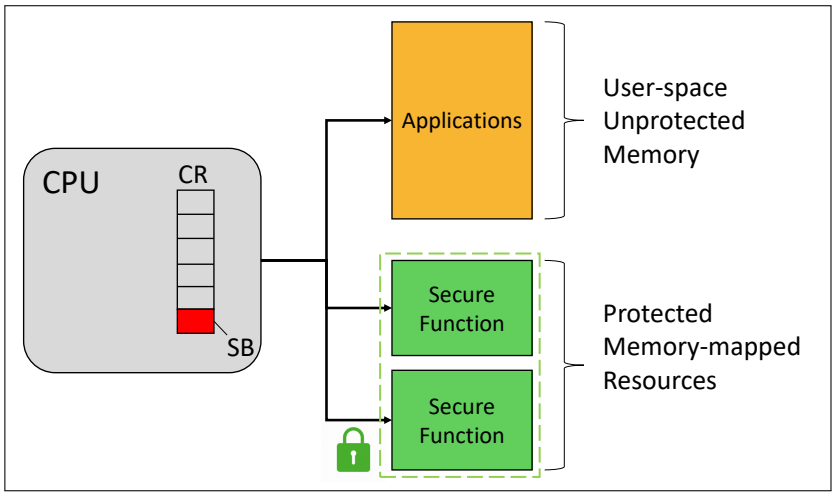

Figure 5.1: Functional scheme of a generic microprocessor system implementing a Trusted Execution Environment. The Security Bit (SB) in the Configuration Register (CR) determines if applications executed from the user-space memory can access the protected resources.

segments and memory mapped peripherals that can only be accessed by trusted software. When the processor is switched to secure mode (i.e., *SB=1*), the applications fetched from the user-space memory are considered trusted, thus secured resources can be accessed.

Working Principle

The presence of scan chains inside the CPU makes the SB controllable by the external world. If the access to the test infrastructure is not protected, an attacker can stop the execution of the processor at any convenient moment, switch it to test mode and shift in the desired data. This allows the attacker to load all the processor flip-flops to predetermined values. This would force the processor to start its execution from a predetermined internal state, potentially from a prohibited state (i.e., a processor state that would never be reachable in functional mode). For instance, by properly setting the right values in the scan flip-flops, an attacker could force the SB to '1' and the program counter to a predetermined address. This, then, would fetch instructions from the unprotected user-space memory. At this point, it would be sufficient for the attacker to previously load a malicious piece of code, which accesses secured resources, into the memory. The value of the program counter could be set to an entry point of the malicious code. This way, the malicious code would be executed in secure mode, making the secured resources accessible. Hence, the security of the system would entirely be annihilated.

As already discussed, protecting the test infrastructure with

scan encryption prevents the attacker from controlling the scan chain content in a predefined way, but it still makes the attacker capable of inserting random bits into the test infrastructure. We adapted the aforementioned attack to a scenario where scan encryption is implemented. For the attack to be successful, the attacker that inserts random values into the processor scan chain must satisfy three requirements:

1. Setting the SB to '1';
2. Loading the program counter with the address of an entry point to the malicious code;
3. Setting the overall state of the microprocessor to a valid state (i.e., the microprocessor does not crash).

It is worth highlighting that each flip-flop in the scan chain can be set independently of the others. For this reason, the probability of setting a specific flip-flop to a fixed value is equal to $\frac{1}{2}$. Knowing this, the probability that the condition 1) is verified is equal to $\frac{1}{2}$. The probability that the condition 2) is verified is equal to $p \times 2^{-N_{PC}}$, where N_{PC} is the number of bits of the program counter and p is the number of entry points to the malicious code. The probability that the condition 3) is verified cannot be analytically computed, due to its complexity.

In order to increase the probability associated to the condition 2), we have increased the number of entry points to the malicious program. This is achieved by writing into the memory a large number of `jump` instructions, which redirect the execution flow to the beginning of the malicious code. At this point, the probability that the condition 2) is verified increases proportionally with the number of `jump` instructions that have been added into the memory. Figure 5.2 shows the principle of the attack.

Attack Implementation

We have implemented the proposed attack on a microprocessor-based system that we have designed. The target system is based on a 32-bit MiniMIPS CPU, which has been modified in order to integrate a TEE. The CPU is connected to an external bus. On the same bus, the user-space unprotected memory is connected, which contains the code executed in normal mode. Secured peripherals are also connected to the system bus. Their programming interface is

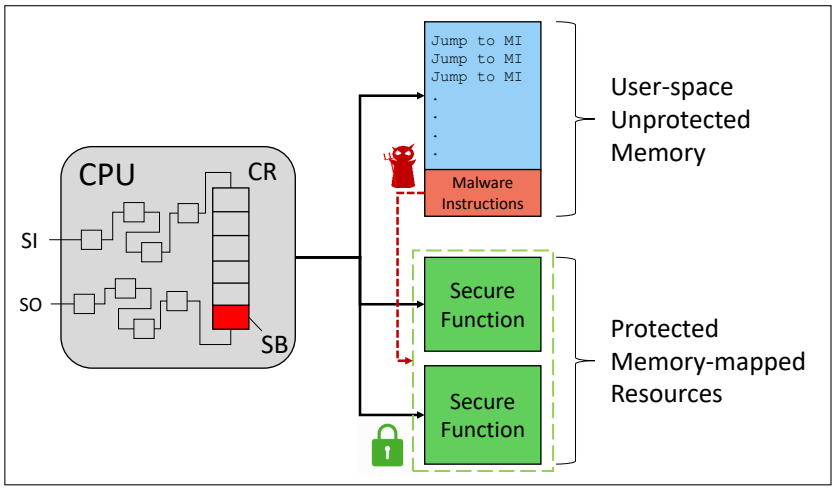

Figure 5.2: Functional scheme of the architecture under attack. The user-space unprotected memory is filled with *jump* instructions redirecting to the malware instructions, which access the protected resources if *SB=1*.

provided by configuration registers that are memory mapped to the address space of the CPU. If *SB=0* and the CPU accesses memory addresses that are mapped to the secured resources, an interrupt is generated and the execution is halted.

We have synthesized the target architecture on a 65nm standard-cell library resorting to Synopsys Design Compiler [89]. Scan chain insertion has automatically been performed using the same tool. Hence, *scan-in* (SI), *scan-out* (SO) and *scan-enable* (SE) pins are added to the CPU. The attack has been simulated resorting to the Mentor ModelSim logic simulator [96]. After the scan chain insertion, we have obtained a unique scan chain traversing all the flip-flops of the CPU, i.e., 1938 flip-flops. The user-space memory has been loaded with the malicious program extended with *jump* instructions. The target address of all the *jump* instructions is the memory location of the malware instruction. The malware instruction is a single *load* instruction that reads the value stored into an address, mapped on the secured resources.

Once the setup is done, the attack is performed through the following procedure:

1. Switching the CPU to test mode (i.e., *SE=1*);
2. A random bit stream is shifted into the scan chain through the scan-in pin;
3. When the scan chain is full (i.e., after 1938 clock cycles) the circuit is switched to functional mode (i.e., *SE=0*);
4. After five clock cycles (i.e., the time needed to execute the malicious program) the system bus is probed. If the CPU is trying to access the secured resource, and the operation is successful, the attack is done. In any

Memory Size	Probability	Attack Time @ 10 MHz
4 MB	0.03%	647.7 ms
8 MB	0.05%	388.6 ms
16 MB	0.06%	323.8 ms
32 MB	0.2%	97.2 ms
64 MB	0.4%	48.6 ms

Table 5.1: Complexity of the scan attack performed on a secure microprocessor based on the Mini-MIPS CPU

other situation, the attempt has failed. Hence, we go back to step 1) for another attempt.

In order to evaluate the success probability of the attack, we have measured how many times the attack was successful out of the total number of attempts. We have bounded the number of attempts to 10^4, according to the computational resources at our disposal. The probability of successfully performing the attack depends on the number of entry points to the malware instruction that are present in the user-space memory. In fact, the higher is the number of entry points, the higher the probability is that the random address inserted into the program counter starts a code execution that is redirected to the malware instruction. Table 5.1 shows the obtained results. The first column reports the size of the user-space memory that has been filled with entry points (i.e. *jump* instructions) to the malware instruction. In the third column, we have reported the average time needed to perform the attack on a physical implementation, clocked at 10 MHz. This value is computed as the average number of attempts needed to succeed with the attack, multiplied by the time needed to perform one attempt. The time needed by the attacker to perform one attempt is equal to

$$T_i = T_{CK} \times (1938 + 5) \tag{5.1}$$

Where T_{CK} is the clock period, 1938 is the number of clock cycles needed to shift the test vector in the scan chain, and 5 is the required number of clock cycles needed to execute the malicious program (i.e. the `jump` instruction plus the malware instruction). At a clock frequency of 10 MHz the required time to perform one attempt is 194.3 μs.

It can be observed that increasing the memory size filled with `jump` instructions leads to a higher probability of success. If the available memory for the malicious program is at least 4 MB, the average attack time is never greater than one second. This means that the present scan attack has

a very low complexity. The easiness of implementation of this attack shows that scan encryption is not sufficient to protect integrated circuits. In fact, leaving the attacker free to insert random data into the test infrastructure exposes the protected device to the present scan attack.

5.3 Scan Encryption with Parity Check

In this section, we present the Scan Encryption with Parity check (SEP) technique, in order to thwart the scan attack presented in the previous section. The key hypothesis that underpins the feasibility of the proposed scan attack, is that symmetric encryption schemes lack of an authentication mechanism, thus the recipient always accepts the received message. With the SEP technique, we propose to enhance scan encryption adding a lightweight integrity check of the data shifted into the test infrastructure. The basic idea consists in encoding, before the encryption, the plaintext test vectors with a publicly known encoding algorithm. The recipient device, after decryption, decodes the obtained test vectors and checks their compliance before applying them to the circuit logic. In the SEP technique, we have chosen to use a parity code coupled with block cipher encryption in order to implement this authentication scheme.

The security of this scheme relies on the following assumption: *an unauthorized user is not capable of crafting a ciphertext, so that, once decrypted, the resulting plaintext is compliant with the desired format*. The paper in [97] reports an attack where the stream cipher encryption used in the GSM protocol is broken due to the application of an error correcting code before encryption. However, we have no knowledge of similar attacks on block cipher implementations. Therefore, we believe that the introduction of redundant bits (in our case, the parity bits) into the test vectors before the encryption process does not jeopardize the overall security.

[97]: Barkan et al. (2008), 'Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication'

In the SEP technique, we resort to parity code computation on each encryption block [98] . Parity decoding is performed in conjunction with the scan input decryption mechanism. An attacker who does not know the encryption key is therefore not able to produce valid encrypted test vectors which successfully pass the parity verification after decryption. As soon as the decryption module detects a wrong parity bit,

[98]: Roth (2006), *Introduction to Coding Theory*

it rises a flag inside the circuit. When the loading of the scan chain is finished and the attacker switches the circuit to functional mode, the presence of the flag makes the circuit turn into a protection state. The SEP technique guarantees the following security requirements:

1. Unauthorized users cannot control or observe the internal state of the circuit. Inheriting the encryption of the scan chain, we ensure that all data that is extracted through the test infrastructure is not understandable by the attacker. Furthermore, after the input decryption, the correctness of the parity bits is checked. This prevents the attacker from inserting test vectors with random content in the scan chain;
2. Exchanged messages through the test infrastructure must not be understandable by third-parties. This property is ensured by the encryption of test data performed by a standard block cipher. Thus, the semantic security of the encryption is guaranteed. Notably, even if the attacker is able to query the internal block cipher and take multiple plaintext/ciphertext pairs (i.e., *known-plaintext attack*), it is not possible to find a correlation between them, hence the secret key.

General Overview of the SEP Technique

For the sake of clarity, we present the implementation of the SEP technique referring to a generic application. The target test infrastructure is based on a single scan chain and its interface is accessible from the outside. We use a scan encryption implementation based on block cipher encryption. Encryption and decryption operations are performed resorting to the SKINNY block cipher which encrypts 64-bit blocks of the plaintext into 64-bit blocks of the ciphertext in 36 clock cycles [99] .

[99]: Beierle et al. (2016), 'The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS'

Parity encoding is performed by the user before encryption of the test vectors. Parity decoding is performed by the SEP architecture after on-chip decryption. The security of the SEP technique is based on the fact that the unauthorized insertion of test data inside the test infrastructure can only result in a random sequence of bits. Hence, the probability for the attacker to send a random message with correct parity bits at the end of each encryption block is negligible. In fact, if

L is the length of the scan chain and b is the block length, the number N of parity bits that must be added to the test vectors is equal to:

$$N = \left\lfloor \frac{L}{b-1} \right\rfloor \tag{5.2}$$

When N parity bits are added, the probability for the attacker to guess a valid ciphertext (i.e., a ciphertext that shows valid parity bits after decrytpion) is 2^{-N}. Referring to the scan attack presented in the previous section, the introduction of the SEP technique leads to a drastically lower attack probability. In fact, if the SEP technique is implemented, the attacker must satisfy at the same time i) all the conditions necessary to achieve the attack, ii) guessing the right value of the parity bits.

Usually, the length of the scan chains leads to test vectors made of a high number of encryption blocks. In that case, the value N may be uselessly high. In the following, we show how a big value of N leads to a non-negligible overhead on the test time. For this reason, it can be convenient for the designer to insert a smaller number of parity bits according to the desired security level. We have proposed an architecture of the SEP module that allows the designer to choose the minimum value of N at design time.

Parity bits are only computed on a subset of equally spaced blocks. If p is the distance between two blocks with parity bits, then the SEP module performs parity checking each p decrypted block. This way, the test time overhead is drastically reduced.

SEP Architecture

The SEP architecture has been implemented on a test infrastructure made of one scan chain. The external interface of the test infrastructure consists in the *scan-in*, *scan-out* and *scan-enable* pins. Figure 5.3 shows a high-level view of all the sub-modules.

The SI input receives the encrypted data. These data are then decrypted by the DEC (DEcryption Cipher) module. The DEC module integrates a SKINNY block cipher that decrypts blocks of 64 bits in parallel. These blocks are processed 36 times by the same round function, hence taking 36 clock cycle

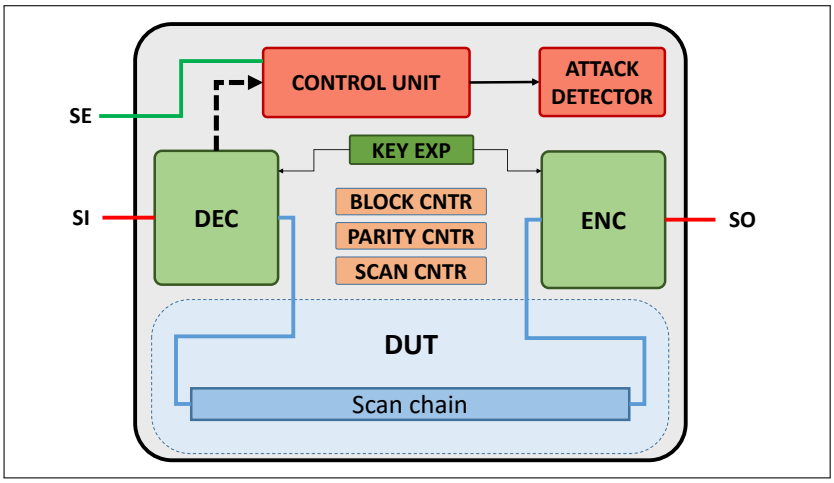

Figure 5.3: Schematics of the SEP architecture. SI and SO pins (in red) carry encrypted test data and responses. The dashed line represents the parity signal, which is asserted by the decryption unit to the control unit.

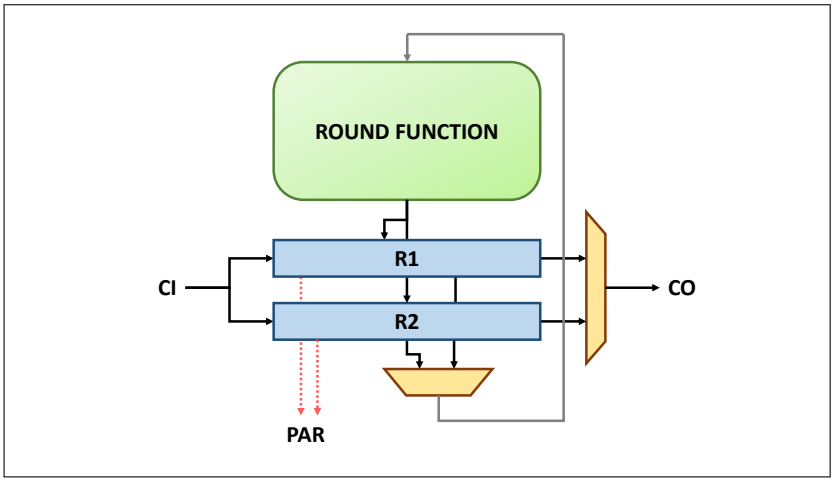

Figure 5.4: High-level schematic of the DEC and ENC units. Parity signals are not present in the ENC unit.

for a single encryption/decryption operation. Inside the DEC module there are two 64-bit registers, $R1$ and $R2$, as shown in Figure 5.4. Both registers can be configured in `shift` or `load` mode. When $R1$ is in `shift` mode, $R2$ is in `load` mode and vice-versa. This means that when the ciphertext stored into $R2$ is processed by the round function, the plaintext stored into $R1$ (decrypted previously) is shifted into the scan chain. At the same time, the next ciphertext block is shifted in $R1$. Both registers are connected to a parity checker. The *parity counter* (`parity_cntr` in Figure 5.3) increments with each decrypted block. At each p block, the Control Unit (CU) is alerted that the DEC operation must take into account the parity bit. When a parity block is decrypted, the last cycle of the `shift` operation is performed with the scan chain disabled (in order not to introduce the parity bit in the scan chain). After that, the CU checks the parity value sent by the DEC module. If the parity is not correct, the CU sends an error message to the Attack Detector (AD).

While the scan chain is loaded with a new test vector, the

response of the previous test vector is shifted out through the SO pin. The ENC (ENcryption Cipher) module performs the encryption of these data using the same principle as in the DEC module. Finally, encrypted data are shifted out via the SO pin. When a parity block is decrypted by the DEC module, the scan chain is put in `hold` mode for one clock cycle. This induces the insertion of a *bubble* inside the response that is being shifted out through the ENC unit. For this reason, test responses have the same additional number of bits as test vectors. For testing purposes, these additional bits can be treated as X-values.

The *key expander* module (`key_exp` in Figure 5.3) takes as input the secret key, stored into a tamper-proof secure memory, and generates the round keys needed at each round of the encryption/decryption operations.

The CU takes as input the SE signal (driven by the tester) plus the control signals coming from the ciphers and the counters. The SE acts as the enable signal of the CU and, as soon as this is reset, the CU guarantees that the scan chain is disabled and the circuit is turned into functional mode.

The AD receives parity error messages from the CU. When the scan chain is totally loaded (condition notified by the *scan counter*), the AD checks if it has received at least one error message. In the affirmative case, an attack is reported to the system. If the SE signal is reset before the scan chain loading is finished, an attack is reported to the system as well. This does not allow the attacker to have a feedback from the system related to a number of parity bits smaller than N.

Security Analysis

When the SEP technique is implemented, the average time required to send a valid test vector to the target device grows exponentially, proportionally to 2^N. We have implemented the MiniMIPS CPU with $N = 31$. In this case, the choice of N was bounded by the scan chain being relatively short. In fact, each test vector is made of 31 encryption blocks. If clocked at 10 MHz, the average time needed to guess a valid test vector in the MiniMIPS CPU is almost 10 months, on average.

If we consider the scan attack on the TEE as our threat model, the probability of performing the attack is the product

Memory Size	Probability	Attack Time @ 10 MHz
4MB	1.4×10^{-13}	1.4×10^{9}s
8MB	2.3×10^{-13}	8.4×10^{8}s
16MB	2.8×10^{-13}	6.9×10^{8}s
32MB	9.3×10^{-13}	2.1×10^{8}s
64MB	1.8×10^{-12}	1.0×10^{8}s

Table 5.2: Complexity of the scan attack on TEE performed on a secure microprocessor based on the MiniMIPS CPU protected with the SEP technique

between the probability of the brute force attack on the parity bits and the probability of success of the original scan attack (without the SEP protection). Table 5.2 shows the scan attack performance if performed on a device protected with the SEP technique. If you compare these figures with the ones in Table 5.1 on page 95, it is possible to appreciate the effectiveness of the SEP technique. The required time to perform the attack goes from the range of tens of milliseconds (without the SEP technique) to years (with the SEP technique).

DfT Flow with SEP Insertion

When the DUT netlist is available, the scan chain (containing a number L of flip-flops) is inserted, and the test sequence (containing a number T of test vectors) is obtained from the ATPG. After that, the designer can choose the desired number N of parity bits, according to the required security level. The expanded test vectors have now $L + N$ bits. Since the decryption and encryption operations are performed per block, the length of each encrypted test vector must be a multiple of the cipher block length b. Now, we assume that the value $L + N$ is a multiple of b. Hence, $L + N = xb$. For $L + N = xb + y$, the scan chain can be completed by y extra flip-flops, allowing the designer to insert observation points and consequently reduce the number of test vectors [100] .

[100]: Da Silva et al. (2017), 'Experimentations on scan chain encryption with PRESENT'

Given the parameters L, b and N, it is possible to make an automated procedure that creates the SEP-compatible test vectors. Let n_b be the number of cipher blocks composing each test vector

$$n_b = \frac{L + N}{b} \tag{5.3}$$

Let p be the distance (in cipher blocks) between two consecutive parity bits on the test vector

$$p = \left\lfloor \frac{n_b}{N} \right\rfloor \tag{5.4}$$

Table 5.3: Benchmarks Characteristics

	SC Length	Area [mm^2]	Test Time [cc]
AES-256	12 736	0.672	4 559 845
3DES	8 808	0.187	687 101
AES-128	7 873	0.368	1 944 877
RSA-1024	16 459	0.463	39 405 239
LEON3	107 518	3.956	11 612 051
MiniMIPS	1 938	0.07	752 331

Table 5.4: SEP Insertion Overhead

	N	Area Overhead [%]	Test Time Overhead [%]
AES-256	33	0.67	0.27
3DES	34	2.40	0.43
AES-128	34	1.22	0.55
RSA-1024	33	0.97	0.23
LEON3	30	0.24	0.03
MiniMIPS	31	13.56	1.63

Since the $\lfloor\ \rfloor$ operation is performed, the actual number of inserted parity bits will be $N_{ins} \geq N$. Notably

$$N_{ins} = \frac{n_b - 1}{p} + 1 \tag{5.5}$$

As mentioned earlier, N_{ins} test responses blocks present an extra X-value bit.

Area and Test Time Overhead

The SEP module has been modeled at the RT-level and wrapped on different benchmark ICs. The ICs have been synthesized and the scan chain has been inserted using the Synopsys Design Compiler suite [89]. The synthesis has been performed on a 65 nm standard-cell library. Table 5.3 shows basic characteristics of the chosen ICs in terms of scan chain (SC) length (expressed in number of flip-flops), area and test time (when traditional test is employed).

The first four circuits in the table are cryptographic processors. LEON3 is a complex SoC for aerospace applications. The MiniMIPS CPU is the same that has been used to implement the scan attack in Chapter 5. The reported test time has been obtained generating test sequences with the Synopsys TetraMax ATPG [90].

Table 5.4 on the preceding page shows the area and test time overhead due to the insertion of the SEP countermeasure. The chosen value of N is also reported. The SEP parameters n_b, p and N_{ins} have been computed resorting to the model reported previously. The number of added parity bits has been established setting the expected attack time, in the case of brute force attack, to about 1 year. The area footprint of the SEP module is almost constant. It slightly grows with the length of the scan chain. This is due to the fact that two counters (i.e., the *scan counter* and the *parity counter* in Figure 5.3) have one more flip-flop when the length of the scan chain doubles. However, related to the footprint of complex SoCs, the overhead is kept very low. The MiniMIPS CPU, in Table 5.4 on the facing page, shows the highest overhead. This must not mislead the interpretation of the results. In fact, we have evaluated the implementation cost of the SEP technique on the MiniMIPS CPU for the sake of exhaustiveness, this being the employed benchmark for the evaluation of the scan attack on the TEE. However, this is a very simple and small CPU that it is very unlikely implemented as a stand-alone CPU. On the other hand, its integration into a far more complex SoC is a more realistic scenario.

Without any security feature, the test time (in clock cycles) is equal to

$$t_{test} = L \times (T + 1) + T \tag{5.6}$$

Adding the SEP technique, the actual test time becomes

$$t^s_{test} = (L + N_{ins}) \times (T + 1) + T + 4b \tag{5.7}$$

The term $4b$ comes from the presence of the block cipher registers at the *scan-in* and *scan-out* of the device. For this reason, loading the first test vector inside the scan chain and downloading the last response will take $L + N_{ins} + 2b$ clock cycles in both cases. Table 5.4 on the preceding page shows that the test time overhead is kept very small in all the benchmarks. The only exception is the MiniMIPS CPU, because of its short scan chain. In fact, in this case one parity bit in each encryption block has been added. These results allow us to remark the importance of carefully optimizing the number of inserted parity bits. As the example of the scan attack on the MiniMIPS CPU has shown, a real attack is way more complex than a brute force attack. Hence, the number of parity bits can be kept lower and the test time overhead can be further reduced.

Discussion

It is worth considering that complex SoCs are the ones that usually have the highest demands in terms of security because of the heterogeneity of their functionality. For this reason, microprocessors with trusted execution environments are widely employed. We have shown that this kind of devices is vulnerable to scan attacks, even if scan encryption is implemented. Thus, we have proposed the SEP module to enhance the overall security at the cost of negligible implementation overhead. The advantages in terms of security are relevant due to the twofold nature of the countermeasure:

1. Unauthorized users cannot read out the content of the test infrastructure nor insert illegal test data. The encryption offered by the ENC unit, right upstream the *scan-out* pin, grants confidentiality of all the content that is extracted by the scan chain. Moreover, random ciphertexts cannot be inserted inside the test infrastructure. The parity bit check assures that sent data have the right format (i.e., the inclusion of a predetermined number of parity bits) and malicious replications of this format cannot be performed because of the encryption function that breaks, as far as we have experimented, every relationship between the ciphertext and the plaintext contents;
2. The authorized user is able to set a completely encrypted communication with the target device, making sniffing attempts useless by both untrusted devices and malicious users. The deployment of a standard block cipher assures that the encryption of the communication has reasonable security against known cryptanalysis.

One limit of the SEP technique is due to the parallel interface of the encryption primitive, which necessarily performs a per-block encryption scheme. If exchanged test data are too short (i.e., $n_b \leq N$), it is impossible to embed the required number of parity bits. For example, when dealing with a test infrastructure equipped with a TAP controller, the test sequences used to load the instructions into the TAP instruction register are very short. These scenarios would require to set the SEP module so that it is effective only when test data are loaded inside test data registers that are sufficiently long.

6 Conclusions

The need for a security-aware testing is a pressing matter for all modern electronic systems. Invasive test infrastructures are mandatory for granting the successful production of defect-free integrated circuits. In most applications, it is not possible to imagine a system without a TAP access anymore. In this thesis, we show that digital security is a major problem affecting all test infrastructures. For this reason, the DfT flow should take security into consideration since early IC development steps. In this thesis, we have shown that many existing countermeasures for test infrastructures are based on the add-on of cryptographic modules for the user authentication. Most of these authentication mechanisms are based on complex cryptographic hardware, representing an unbearable cost outside of certain specific niche markets. In the same way that DfT has experienced a process of automation in recent decades, which has allowed its massive use in all market segments, we want to emphasize that the need for a secure DfT should also know such a spread. Although there are now numerous proposals in the literature in this direction, the scenario is still too fragmented, and composed of techniques that offer partial protection and/or whose security is difficult to prove. We identified the *scan encryption* as a very promising technique for easily making test infrastructures secure. Scan encryption is based on the add-on of cryptographic hardware whose cost can be kept marginal according to the designer need. Its deployment is extremely simple and easy to automatize. The DfT designer establish the secret keys that are used by the device, and the whole test flow is encrypted accordingly. This hinders both unauthorized access from malicious users and information leakage in the external and internal communication channels.

In this thesis, we have thoroughly analyzed scan encryption techniques and tried to solve the problems that have been identified. At first, we have inspected the state-of-the-art techniques, identifying two categories: *stream based* and *block based* scan encryption. While block based scan encryption had already been the subject of a careful analysis and extensive

experimental activity, stream based scan encryption had been proposed in numerous works, but never subjected to a thorough security analysis. For this reason, we have proposed new scan encryption implementations based on stream ciphers, fixing the security problems identified in the state-of-the-art. The proposed stream based scan encryption is based on the random generation of a different IV at the beginning of each communication between the user and the device. Although this implies higher implementation costs, the stream based and block based approaches have been brought to the same security level, and they can both be safely considered for the deployment of a secure DfT. We have extensively compared the new stream based scan encryption and the already existent block based scan encryption. We have highlighted pros and cons of the two solutions, in order to provide a guideline for the designers that are interested in employing scan encryption for securing their DfT.

Scan encryption, both in its stream based and block based implementation, is able to thwart most of the attacks involving test infrastructures. In fact, both controllability and observability on the internal resources is drastically hindered. The attacker has no control over the decryption operation that is performed at the scan-in port. Thus, his/her only capability is inserting random data inside the test infrastructure. On the other side, data encryption guarantees the confidentiality of test data outside the boundaries of the target device. Thus, all attempts of sniffing and illegal access to the scan-out pin do not lead to any threat. Despite the extensive security offered by scan encryption, its lack of a data integrity check mechanism makes it potentially vulnerable to unwanted accesses. In this thesis, we have explored possible scenarios in which a malicious user can exploit the capability of inserting random data in the test infrastructure to perform an attack. We have identified a category of attacks, in which the attacker exploits the internal scan chains in order to force a very limited number of flip-flops to a specific value. In this case, the scan encryption countermeasure is not enough to avoid a scenario. We have proposed an example attack targeting the Trusted Execution Environment of a microprocessor. If the attacker appropriately sets the microprocessor memory, it is possible to trigger the execution of malicious code through the insertion of random data in the scan chains. For this reason, we have proposed an enhancement of the scan encryption including a very lightweight integrity check mechanism. In

this technique, we propose to include a format rule to the plaintext test data that the target device can easily check after decryption. If these rules are respected, the received data are valid, otherwise, they are considered malicious data sent by an attacker. The security of this technique lies on the fact that an attacker cannot produce a valid message that would result in the correct format after decryption, without knowing the scan encryption key. We implemented this technique adding parity bits in the test data. At the cost of a little overhead in test time and area, scan encryption acquires a security property that makes it robust against any kind of unauthorized access.

We believe that scan encryption is a very promising technique for protecting test infrastructures. Its characteristics make it suitable for providing a security-aware DfT even in low-cost devices. Its flexibility and the variety of implementations, are characteristics that allow to adapt the countermeasure to different design needs. We also think that the development of a security-aware test standard is unavoidable in the future, and scan encryption could be a promising starting point for reasoning about this.

6.1 Future Perspectives

If we extrapolate the scan encryption from its application as a security-aware DfT technique, we obtain a security method that can be applied to any communication protocol involving a serial interface on the IC. Due to its low implementation cost, symmetric encryption is a valid option for providing a good security level in domains where the implementation costs are a critical issue, such as small devices for the Internet of Things (IoT).

We have found an alternative application of the scan encryption technique in the domain of *intermittent computing systems*. These are embedded systems whose power supply is not stable over time, usually because it relies on *energy harvesting* techniques. For this reason, the system must be ready to be interrupted at any time, and it must be able to recover its computation state at the power-on incurring in minimal loss of information [101–104]. Numerous techniques proposed in the literature aim at providing an efficient mechanism to store the computation state of the CPU when the

power supply is going to be interrupted soon, and recover it as soon as the power supply is available again [105–108]. Some of these techniques rely on the scan chains in order to quickly access the CPU's flip-flops and download their content on a target non-volatile memory (NVM) [109] . In this scenario, it is necessary to properly secure the content of the scan chains while it is stored into the target NVM. In fact, target NVMs in this kind of applications are usually based on *ferromagnetic RAMs* (FeRAM), which can be easily tampered with relying on low-cost physical attacks [110, 111]. We think that the stream based scan encryption is a suitable proposal for protecting against this kind of procedures. A preliminary work has been done in this direction. [112, 113] When the system is going to be turned down, the content of the scan chain is shifted out and encrypted by a stream cipher. Furthermore, a MAC is appended to the end of the message as an integrity mechanisms. This way, the NVM stores encrypted data that cannot be tampered with by the attackers. When the system is ready to recover its computation state, the content of the NVM is decrpyted and shifted back into the scan chains. Moreover, the MAC signature is checked in order to prove the integrity of the information. The key generation can be managed using a PUF.

[109]: Hager et al. (2017), 'A scan-chain based state retention methodology for IoT processors operating on intermittent energy'

[112]: Valea et al. (2018), 'SI ECCS: SECure context saving for IoT devices'

[113]: Valea et al. (2019), 'Providing Confidentiality and Integrity in Ultra Low Power IoT Devices'

In this thesis, we have proposed to enhance the scan encryption technique adding a parity coding to the plaintext before its encryption (i.e., the SEP technique). After decryption, the value of the parity bits is checked and, if their value is incorrect, this indicates a corruption of the encrypted data and/or an unauthorized user sending a message that has not been encrypted with the correct key. Apparently, the SEP technique can be seen as an integrity technique addressing the same threat model as integrity techniques based on hash functions and message authentication codes. In cryptography, we also reckon the existence of *authenticated encryption* (AE) techniques, which provide both data encryption and their signature based on a MAC code. Further research could be pointed out to provide an extensive comparison between the SEP technique and AE techniques based on state-of-the-art cryptographic functions. Parity bits coding has been chosen for the implementation of the SEP technique because of its easiness of use and its popularity in the design of integrated digital systems. However, it would be interesting to explore other possibilities that could even lead to a smaller cost of implementation. For instance, adding constant bits

to a specific value inside the test vectors could be equally effective, as far as the number of added bits is big enough in order to counteract brute force attacks. More generally speaking, we believe that further research should be done in order to better evaluate the security of symmetric encryption schemes against the presence of known patterns inside the plaintext.

Scientific Contributions

Here we enlist the scientific publications and the dissemination activities related to the topic of this thesis.

Publications on Scientific Journals

1. **"A Survey on Security Threats and Countermeasures in IEEE Test Standards"**
   E. Valea, M. Da Silva, G. Di Natale, M. Flottes, B. Rouzeyre
   IEEE Design & Test, Volume 36, pp 95-116. June 2019.
   DOI: 10.1109/MDAT.2019.2899064

2. **"Stream vs block ciphers for scan encryption"**
   E. Valea, M. Da Silva, M. Flottes, G. Di Natale, B. Rouzeyre
   Microelectronics Journal, Elsevier, Volume 86, pp 65-76. April 2019.
   DOI: 10.1016/j.mejo.2019.02.019

Publications on the Proceedings of International Conferences

1. **"Encryption-Based Secure JTAG"**
   E. Valea, M. Da Silva, M. Flottes, G. Di Natale, B. Rouzeyre
   DDECS 2019: IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems.
   Cluj-Napoca, Romania, 24-26 April 2019.
   DOI: 10.1109/DDECS.2019.8724654

2. **"Encryption of test data: which cipher is better?"**
   M. Da Silva, E. Valea, M. Flottes, S. Dupuis, G. Di Natale, B. Rouzeyre
   PRIME 2018: 14th Conference on Ph.D. Research in Microelectronics and Electronics.
   Prague, Czech Republic, 2-5 July 2018.
   DOI: 10.1109/PRIME.2018.8430366

Publications on the Proceedings of International Workshops

1. **"A Comprehensive Approach to a Trusted Test Infrastructure"**
   M. Merandat, V. Reynaud, E. Valea, J. Quevremont, N. Valette, P. Maistri, R. Leveugle, M. Flottes, S. Dupuis, B. Rouzeyre, G. Di Natale
   IVSW 2019: IEEE 4th International Verification and Security Workshop.
   Rhodes Island, Greece, 1-3 July 2019.
   DOI: 10.1109/IVSW.2019.8854428

2. **"A New Secure Stream Cipher for Scan Chain Encryption"**
   M. Da Silva, E. Valea, M. Flottes, S. Dupuis, G. Di Natale, B. Rouzeyre
   IVSW 2018: IEEE 3rd International Verification and Security Workshop.
   Costa Brava, Spain, 2-4 July 2018.
   DOI: 10.1109/IVSW.2018.8494852

Publications on the Proceedings of National Conferences

1. **"Encryption Techniques for Test Infrastructures"**
   E. Valea, M. Flottes, G. Di Natale, B. Rouzeyre
   GDR SOC^2 2019: Colloque GDR SoC^2.
   Montpellier, France, 19-21 June 2019.

2. **"Stream cipher-based scan encryption in test standards"**
   M. Da Silva, E. Valea, M. Flottes, G. Di Natale, B. Rouzeyre
   GDR SOC^2 2018: Colloque GDR SoC^2.
   Paris, France, 13-15 June 2018.

Other Dissemination Activities

1. **"Stream Cipher Based Encryption in IEEE Test Standards"**
   E. Valea, M. Flottes, G. Di Natale, B. Rouzeyre
   TRUDEVICE 2019: 8th Workshop on Trustworthy Manufacturing and Utilization of Secure Devices.
   Baden Baden, Germany, 30-31 May 2019.

2. **"Encryption Techniques for Test Infrastructures"**
   E. Valea, M. Flottes, G. Di Natale, B. Rouzeyre
   PhD Forum @ ETS 2019: IEEE European Test Symposium.
   Baden Baden, Germany, 27-31 May 2019.

3. **“Test Standards and Security”**
   E. Valea, M. Flottes, G. Di Natale, B. Rouzeyre
   SETS 2018: South European Test Seminar.
   Kaprun, Austria, 20-22 February 2018.

Bibliography

Here are the references in citation order.

[1] J. Andrews. 'IEEE Standard Boundary Scan 1149.1 An Introduction'. In: *Electro International, 1991*. 1991, pp. 522–527 (cited on pages vi, 7, 8).

[2] 'IEEE Standard Testability Method for Embedded Core-based Integrated Circuits'. In: *IEEE Std 1500-2005* (Aug. 2005), pp. 1–136. DOI: 10.1109/IEEESTD.2005.96465 (cited on pages vi, 7, 9).

[3] 'IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device'. In: *IEEE Std 1687-2014* (Dec. 2014), pp. 1–283. DOI: 10.1109/IEEESTD.2014.6974961 (cited on pages vi, 7, 11).

[4] K. Rosenfeld and R. Karri. 'Attacks and Defenses for JTAG'. In: *IEEE Design & Test of Computers* 27.1 (Jan. 2010), pp. 36–47. DOI: 10.1109/MDT.2010.9 (cited on pages xii, 26, 28, 45–47, 59, 63).

[5] K. Rosenfeld and R. Karri. 'Security-aware SoC test access mechanisms'. In: *29th VLSI Test Symposium*. May 2011, pp. 100–104. DOI: 10.1109/VTS.2011.5783765 (cited on pages xii, 59, 63).

[6] S. Kan, J. Dworak, and J. G. Dunham. 'Echeloned IJTAG data protection'. In: *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. Dec. 2016, pp. 1–6. DOI: 10.1109/AsianHOST.2016.7835558 (cited on pages xiii, 27, 41, 49, 59, 63).

[7] Laung-Terng Wang, Xiaoqing Wen, and Khader S. Abdel-Hafez. *VLSI Test Principles and Architectures*. 1st ed. Morgan Kaufmann, 2006 (cited on pages 2, 4, 6).

[8] Michael L. Bushnell and Vishwani D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. 1st ed. Springer, Boston, MA, 2002 (cited on pages 2, 4).

[9] M. Portolan. 'Automated Testing Flow: the Present and the Future'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019), pp. 1–1 (cited on page 3).

[10] M. Psarakis et al. 'Microprocessor Software-Based Self-Testing'. In: *IEEE Design & Test of Computers* 27.3 (2010), pp. 4–19 (cited on page 3).

[11] International Standard Organization. *ISO 26262-1:2018. Road vehicles — Functional safety.* https://www.iso.org/standard/68383.html. Accessed: July 17, 2020 (cited on page 3).

[12] C. Barnhart et al. 'OPMISR: the foundation for compressed ATPG vectors'. In: *Proceedings International Test Conference 2001 (Cat. No.01CH37260)*. 2001, pp. 748–757 (cited on page 6).

[13] J. Rajski et al. 'Embedded deterministic test'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23.5 (2004), pp. 776–792 (cited on page 6).

[14] 'IEEE Standard for Test Access Port and Boundary-Scan Architecture'. In: *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)* (May 2013), pp. 1–444. DOI: 10.1109/IEEESTD.2013.6515989 (cited on page 7).

[15] A. Tšertov et al. 'A suite of IEEE 1687 benchmark networks'. In: *2016 IEEE International Test Conference (ITC)*. 2016, pp. 1–10 (cited on page 11).

[16] A. Damljanovic et al. 'Post-Silicon Validation of IEEE 1687 Reconfigurable Scan Networks'. In: *2019 IEEE European Test Symposium (ETS)*. 2019, pp. 1–6 (cited on page 11).

[17] R. Cantoro et al. 'A Novel Sequence Generation Approach to Diagnose Faults in Reconfigurable Scan Networks'. In: *IEEE Transactions on Computers* 69.1 (2020), pp. 87–98 (cited on page 11).

[18] R. Cantoro et al. 'A New Technique to Generate Test Sequences for Reconfigurable Scan Networks'. In: *2018 IEEE International Test Conference (ITC)*. 2018, pp. 1–9 (cited on page 11).

[19] R. Cantoro et al. 'A Semi-Formal Technique to Generate Effective Test Sequences for Reconfigurable Scan Networks'. In: *2018 IEEE International Test Conference in Asia (ITC-Asia)*. 2018, pp. 55–60 (cited on page 11).

[20] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 'Differential Power Analysis'. In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397 (cited on page 13).

[21] D. Hely et al. 'Scan design and secure chip [secure IC testing]'. In: *Proceedings. 10th IEEE International On-Line Testing Symposium*. 2004, pp. 219–224 (cited on page 13).

[22] Bo Yang, Kaijie Wu, and Ramesh Karri. 'Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard'. In: *2004 International Conference on Test*. Oct. 2004, pp. 339–344. DOI: 10.1109/TEST.2004.1386969 (cited on pages 13, 17, 20).

[23] J. Da Rolt et al. 'Test Versus Security: Past and Present'. In: *IEEE Transactions on Emerging Topics in Computing* 2.1 (Mar. 2014), pp. 50–62. DOI: 10.1109/TETC.2014.2304492 (cited on page 13).

[24] E. Valea et al. 'A Survey on Security Threats and Countermeasures in IEEE Test Standards'. In: *IEEE Design & Test* 36.3 (June 2019), pp. 95–116. DOI: 10.1109/MDAT.2019.2899064 (cited on page 13).

[25] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. 1st ed. Springer-Verlag Berlin Heidelberg, 2002 (cited on page 14).

[26] I. Verbauwhede, P. Schaumont, and H. Kuo. 'Design and performance testing of a 2.29-GB/s Rijndael processor'. In: *IEEE Journal of Solid-State Circuits* 38.3 (2003), pp. 569–572 (cited on page 14).

[27] S. Mangard, M. Aigner, and S. Dominikus. 'A highly regular and scalable AES hardware architecture'. In: *IEEE Transactions on Computers* 52.4 (2003), pp. 483–491 (cited on page 15).

[28] *Maestra Comprehensive Guide to Satellite TV Testing*. http://www.angelfire.com/. Accessed: July 17, 2020 (cited on page 15).

[29] B. Yang, K. Wu, and R. Karri. 'Secure Scan: A Design-for-Test Architecture for Crypto Chips'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.10 (Oct. 2006), pp. 2287–2293. DOI: 10.1109/TCAD.2005.862745 (cited on pages 15, 17, 20).

[30] Federal Information Processing Standards. *Data Encryption Standard*. 1977 (cited on page 16).

[31] Yu Liu, Kaijie Wu, and Ramesh Karri. 'Scan-Based Attacks on Linear Feedback Shift Register Based Stream Ciphers'. In: *ACM Trans. Des. Autom. Electron. Syst.* 16.2 (Apr. 2011). DOI: 10.1145/1929943.1929952 (cited on pages 17, 20).

[32] J. Da Rolt et al. 'A scan-based attack on Elliptic Curve Cryptosystems in presence of industrial Design-for-Testability structures'. In: *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. Oct. 2012, pp. 43–48. DOI: 10.1109/DFT.2012.6378197 (cited on pages 17, 20).

[33] J. Da Rolt et al. 'New security threats against chips containing scan chain structures'. In: *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*. June 2011, pp. 110–110. DOI: 10.1109/HST.2011.5955005 (cited on page 21).

[34] J. DaRolt et al. 'Scan Attacks and Countermeasures in Presence of Scan Response Compactors'. In: *2011 Sixteenth IEEE European Test Symposium (ETS)*. 2011, pp. 19–24 (cited on page 21).

[35] J. Da Rolt et al. 'Are advanced DfT structures sufficient for preventing scan-attacks?' In: *2012 IEEE 30th VLSI Test Symposium (VTS)*. Apr. 2012, pp. 246–251. DOI: 10.1109/VTS.2012.6231061 (cited on page 21).

[36] A. Das et al. 'Security Analysis of Industrial Test Compression Schemes'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32.12 (Dec. 2013), pp. 1966–1977. DOI: 10.1109/TCAD.2013.2274619 (cited on page 21).

[37] S. S. Ali et al. 'New scan-based attack using only the test mode'. In: *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC)*. 2013, pp. 234–239 (cited on page 21).

[38] S. S. Ali et al. 'Novel Test-Mode-Only Scan Attack and Countermeasure for Compression-Based Scan Architectures'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.5 (May 2015), pp. 808–821. DOI: 10.1109/TCAD.2015.2398423 (cited on page 21).

[39] Leonid Azriel, Ran Ginosar, and Avi Mendelson. 'Revealing On-Chip Proprietary Security Functions with Scan Side Channel Based Reverse Engineering'. In: *Proceedings of the on Great Lakes Symposium on VLSI 2017*. GLSVLSI '17. Banff, Alberta, Canada: Association for Computing Machinery, 2017, pp. 233–238. DOI: 10.1145/3060403.3060464 (cited on page 21).

[40] Leonid Azriel, Ran Ginosar, and Avi Mendelson. *Exploiting the Scan Side Channel for Reverse Engineering of a VLSI Device*. https://pdfs.semanticscholar.org/b5e7/d14f3fdbc7a62fe0009beb6a8be21e2582d1.pdf. Accessed: July 17, 2020. 2016 (cited on page 21).

[41] Ing. M.F. Breeuwsma. 'Forensic imaging of embedded systems using JTAG (boundary-scan)'. In: *Digital Investigation* 3.1 (2006), pp. 32–42. DOI: https://doi.org/10.1016/j.diin.2006.01.003 (cited on page 22).

[42] Svein Willassen. 'Forensic Analysis of Mobile Phone Internal Memory'. In: *Advances in Digital Forensics*. Ed. by Mark Pollitt and Sujeet Shenoi. Boston, MA: Springer US, 2005, pp. 191–204 (cited on page 22).

[43] Felix Domke. 'Blackbox JTAG Reverse Engineering'. In: 2009 (cited on page 22).

[44] Sergei Skorobogatov and Christopher Woods. 'Breakthrough Silicon Scanning Discovers Backdoor in Military Chip'. In: *Cryptographic Hardware and Embedded Systems – CHES 2012*. Ed. by Emmanuel Prouff and Patrick Schaumont. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 23–40 (cited on page 23).

[45] Senrio. *JTAG explained (finally!): Why "IoT" software security engineers and manufacturers should care*. https://blog.senr.io/blog/jtag-explained. Accessed: July 17, 2020. 2018 (cited on page 23).

[46] F. Majeric, B. Gonzalvo, and L. Bossuet. 'JTAG Combined Attack - Another Approach for Fault Injection'. In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. Nov. 2016, pp. 1–5. DOI: 10.1109/NTMS.2016.7792458 (cited on page 23).

[47] Y. Zorian. 'A distributed BIST control scheme for complex VLSI devices'. In: *Digest of Papers Eleventh Annual 1993 IEEE VLSI Test Symposium*. Apr. 1993, pp. 4–9. DOI: 10.1109/VTEST.1993.313316 (cited on page 24).

[48] J. Dworak et al. 'Don't forget to lock your SIB: hiding instruments using P1687'. In: *2013 IEEE International Test Conference (ITC)*. Sept. 2013, pp. 1–10. DOI: 10.1109/TEST.2013.6651903 (cited on pages 24, 25, 38–41).

[49] Altera. *Protecting the FPGA Design From Common Threats*. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01111-anti-tamper.pdf. 2010 (cited on page 27).

[50] R. Elnaggar, R. Karri, and K. Chakrabarty. 'Securing IJTAG against data-integrity attacks'. In: *2018 IEEE 36th VLSI Test Symposium (VTS)*. Apr. 2018, pp. 1–6. DOI: 10.1109/VTS.2018.8368642 (cited on pages 27, 28, 47, 49).

[51] Oliver Kömmerling and Markus G. Kuhn. 'Design Principles for Tamper-resistant Smartcard Processors'. In: *Proceedings of the USENIX Workshop on Smartcard Technology*. WOST'99. Chicago, Illinois: USENIX Association, 1999 (cited on page 31).

[52] F. Novak and A. Biasizzo. 'Security Extension for IEEE Std 1149.1'. In: *Journal of Electronic Testing* 22.3 (June 2006), pp. 301–303 (cited on pages 33, 34).

[53] G. Chiu and J. C. Li. 'A Secure Test Wrapper Design Against Internal and Boundary Scan Attacks for Embedded Cores'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.1 (Jan. 2012), pp. 126–134. DOI: 10.1109/TVLSI.2010.2089071 (cited on pages 33, 34).

[54] K. Park et al. 'JTAG Security System Based on Credentials'. In: *Journal of Electronic Testing* 26.5 (Oct. 2010), pp. 549–557. DOI: 10.1007/s10836-010-5170-y (cited on page 35).

[55] C. Clark. 'Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments'. In: *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. June 2010, pp. 19–24. DOI: 10.1109/HST.2010.5513119 (cited on pages 35, 36, 39, 40, 42, 43).

[56] A. Das et al. 'PUF-based secure test wrapper design for cryptographic SoC testing'. In: *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2012, pp. 866–869. DOI: 10.1109/DATE.2012.6176618 (cited on page 35).

[57] R. F. Buskey and B. B. Frosik. 'Protected JTAG'. In: *2006 International Conference on Parallel Processing Workshops (ICPPW'06)*. Aug. 2006, 8 pp.–414. DOI: 10.1109/ICPPW.2006.65 (cited on pages 35, 36).

[58] A. Das et al. 'Secure JTAG Implementation Using Schnorr Protocol'. In: *Journal of Electronic Testing* 29.2 (Apr. 2013), pp. 193–209. DOI: 10.1007/s10836-013-5369-9 (cited on pages 36, 46).

[59] H. Liu and V. D. Agrawal. 'Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key'. In: *2015 IEEE 24th Asian Test Symposium (ATS)*. Nov. 2015, pp. 91–96. DOI: 10.1109/ATS.2015.23 (cited on pages 38, 39).

[60] N. Satheesh et al. 'Securing IEEE 1687 Standard On-chip Instrumentation Access Using PUF'. In: *2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*. Dec. 2016, pp. 56–61. DOI: 10.1109/iNIS.2016.024 (cited on pages 38, 39).

[61] S. Gupta et al. 'Mitigating simple power analysis attacks on LSIB key logic'. In: *2017 IEEE North Atlantic Test Workshop (NATW)*. 2017, pp. 1–6 (cited on page 39).

[62] R. Baranowski, M. A. Kochte, and H. Wunderlich. 'Fine-Grained Access Management in Reconfigurable Scan Networks'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.6 (June 2015), pp. 937–946. DOI: 10.1109/TCAD.2015.2391266 (cited on pages 39, 40, 42–45).

[63] A. Zygmontowicz et al. 'Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687 network'. In: *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2014, pp. 1–6. DOI: 10.7873/DATE.2014.208 (cited on pages 40, 41).

[64] J. Backer, D. Hély, and R. Karri. 'Secure design-for-debug for Systems-on-Chip'. In: *2015 IEEE International Test Conference (ITC)*. Oct. 2015, pp. 1–8. DOI: 10.1109/TEST.2015.7342418 (cited on pages 42, 43).

[65] L. Pierce and S. Tragoudas. 'Enhanced Secure Architecture for Joint Action Test Group Systems'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21.7 (July 2013), pp. 1342–1345. DOI: 10.1109/TVLSI.2012.2208209 (cited on pages 42, 43).

[66] P. Raiola et al. 'Detecting and Resolving Security Violations in Reconfigurable Scan Networks'. In: *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. 2018, pp. 91–96 (cited on page 45).

[67] N. Lylina et al. 'Security Compliance Analysis of Reconfigurable Scan Networks'. In: *2019 IEEE International Test Conference (ITC)*. 2019, pp. 1–9 (cited on page 45).

[68] M. A. Kochte, R. Baranowski, and H. Wunderlich. 'Trustworthy reconfigurable access to on-chip infrastructure'. In: *2017 International Test Conference in Asia (ITC-Asia)*. Sept. 2017, pp. 119–124. DOI: 10.1109/ITC-ASIA.2017.8097125 (cited on page 45).

[69] J. Dworak et al. 'Board security enhancement using new locking SIB-based architectures'. In: *2014 International Test Conference*. Oct. 2014, pp. 1–10. DOI: 10.1109/TEST.2014.7035355 (cited on page 46).

[70] R. Baranowski, M. A. Kochte, and H. Wunderlich. 'Access Port Protection for Reconfigurable Scan Networks'. In: *Journal of Electronic Testing* 30.6 (Dec. 2014), pp. 711–723 (cited on pages 48, 49).

[71] Rafal Baranowski, Michael A. Kochte, and Hans-Joachim Wunderlich. 'Reconfigurable Scan Networks: Modeling, Verification, and Optimal Pattern Generation'. In: *ACM Trans. Des. Autom. Electron. Syst.* 20.2 (Mar. 2015). DOI: 10.1145/2699863 (cited on page 49).

[72] M. A. Kochte et al. 'Formal verification of secure reconfigurable scan network infrastructure'. In: *2016 21th IEEE European Test Symposium (ETS)*. 2016, pp. 1–6 (cited on page 49).

[73] P. Raiola et al. 'On Secure Data Flow in Reconfigurable Scan Networks'. In: *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2019, pp. 1016–1021 (cited on page 49).

[74] A. Atteya et al. 'Online prevention of security violations in reconfigurable scan networks'. In: *2018 IEEE 23rd European Test Symposium (ETS)*. 2018, pp. 1–6 (cited on page 49).

[75] M. A. Kochte et al. 'Specification and verification of security in reconfigurable scan networks'. In: *2017 22nd IEEE European Test Symposium (ETS)*. 2017, pp. 1–6 (cited on page 49).

[76] X. Ren et al. 'IC Protection Against JTAG-Based Attacks'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38.1 (Jan. 2019), pp. 149–162. DOI: 10.1109/TCAD.2018.2802866 (cited on pages 49, 50).

[77] X. Ren, R. D. S. Blanton, and V. G. Tavares. 'Detection of IJTAG attacks using LDPC-based feature reduction and machine learning'. In: *2018 IEEE 23rd European Test Symposium (ETS)*. 2018, pp. 1–6 (cited on page 50).

[78] X. Ren, R. D. Blanton, and V. G. Tavares. 'A Learning-Based Approach to Secure JTAG Against Unseen Scan-Based Attacks'. In: *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2016, pp. 541–546 (cited on page 50).

[79] X. Ren, V. G. Tavares, and R. D. S. Blanton. 'Detection of illegitimate access to JTAG via statistical learning in chip'. In: *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2015, pp. 109–114 (cited on page 50).

[80] C. De Canniere and B. Preneel. *TRIVIUM Specifications*. https://www.ecrypt.eu.org/stream/e2-trivium.html. 2005 (cited on pages 54, 67).

[81] M. Da Silva et al. 'Scan chain encryption for the test, diagnosis and debug of secure circuits'. In: *2017 22nd IEEE European Test Symposium (ETS)*. May 2017, pp. 1–6. DOI: 10.1109/ETS.2017.7968248 (cited on page 57).

[82] M. Da Silva et al. 'Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38.3 (Mar. 2019), pp. 538–550. DOI: 10.1109/TCAD.2018.2818722 (cited on pages 57, 78, 79).

[83] M. Da Silva et al. 'A New Secure Stream Cipher for Scan Chain Encryption'. In: *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*. 2018, pp. 68–73 (cited on page 61).

[84] E. Valea et al. 'Encryption-Based Secure JTAG'. In: *2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*. 2019, pp. 1–6 (cited on page 61).

[85] Emanuele Valea et al. 'Stream vs block ciphers for scan encryption'. In: *Microelectronics Journal* 86 (2019), pp. 65–76. DOI: https://doi.org/10.1016/j.mejo.2019.02.019 (cited on page 61).

[86] M. Merandat et al. 'A Comprehensive Approach to a Trusted Test Infrastructure'. In: *2019 IEEE 4th International Verification and Security Workshop (IVSW)*. 2019, pp. 43–48 (cited on page 61).

[87] Giorgio Di Natale et al. 'Manufacturing Testing and Security Countermeasures'. In: *Hardware Security and Trust*. Ed. by Springer International Publishing. Springer International Publishing, 2017. Chap. 7, pp. 127–148 (cited on page 66).

[88] *DesignWare True Random Number Generator Core*. https://www.synopsys.com/dw/ipdir.php?ds=security-random-number-generator. Accessed: July 17, 2020 (cited on pages 68, 75, 80).

[89] *Design Compiler®*. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-nxt.html. Accessed: July 17, 2020 (cited on pages 69, 94, 102).

[90] *TetraMAX*. https://www.synopsys.com/support/training/signoff/tmax1-fcd.html. Accessed: July 17, 2020 (cited on pages 69, 81, 102).

[91] A. Schubert and W. Anheier. 'On Random Pattern Testability of Cryptographic VLSI Cores'. In: *Journal of Electronic Testing* 16.3 (June 2000), pp. 185–192. DOI: 10.1023/A:1008378912411 (cited on page 82).

[92] G. D. Natale et al. 'Self-Test Techniques for Crypto-Devices'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18.2 (Feb. 2010), pp. 329–333. DOI: 10.1109/TVLSI.2008.2010045 (cited on page 82).

[93] M. Doulcier, M. -. Flottes, and B. Rouzeyre. 'AES-Based BIST: Self-Test, Test Pattern Generation and Signature Analysis'. In: *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*. 2008, pp. 314–321 (cited on page 82).

[94] B. Ngabonziza et al. 'TrustZone Explained: Architectural Features and Use Cases'. In: *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. Nov. 2016, pp. 445–451. DOI: 10.1109/CIC.2016.065 (cited on page 91).

[95] *ARM Security Technology. Building a Secure System using TrustZone Technology*. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html. Accessed: July 17, 2020 (cited on page 91).

[96] *ModelSim®*. https://www.mentor.com/products/fv/modelsim/. Accessed: July 17, 2020 (cited on page 94).

[97] Elad Barkan, Eli Biham, and Nathan Keller. 'Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication'. In: *J. Cryptol.* 21.3 (Mar. 2008), pp. 392–429. DOI: 10.1007/s00145-007-9001-y (cited on page 96).

[98] Ron Roth. *Introduction to Coding Theory*. New York, NY, USA: Cambridge University Press, 2006 (cited on page 96).

[99] Christof Beierle et al. 'The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS'. In: *Advances in Cryptology – CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 123–153 (cited on page 97).

[100] M. Da Silva et al. 'Experimentations on scan chain encryption with PRESENT'. In: *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*. July 2017, pp. 45–50. DOI: 10.1109/IVSW.2017.8031543 (cited on page 101).

[101] G. Berthou et al. 'Peripheral state persistence for transiently-powered systems'. In: *2017 Global Internet of Things Summit (GIoTS)*. 2017, pp. 1–6 (cited on page 107).

[102] Hrishikesh Jayakumar et al. 'QuickRecall: A HW/SW Approach for Computing across Power Cycles in Transiently Powered Computers'. In: *J. Emerg. Technol. Comput. Syst.* 12.1 (Aug. 2015). DOI: 10.1145/2700249 (cited on page 107).

[103] Matthew Hicks. 'Clank: Architectural Support for Intermittent Computation'. In: *SIGARCH Comput. Archit. News* 45.2 (June 2017), pp. 228–240. DOI: 10.1145/3140659.3080238 (cited on page 107).

[104] Z. Ghodsi, S. Garg, and R. Karri. 'Optimal checkpointing for secure intermittently-powered IoT devices'. In: *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2017, pp. 376–383 (cited on page 107).

[105] Z. Liu and V. Kursun. 'New MTCMOS Flip-Flops with Simple Control Circuitry and Low Leakage Data Retention Capability'. In: *2007 14th IEEE International Conference on Electronics, Circuits and Systems*. 2007, pp. 1276–1279 (cited on page 108).

[106] Jean-Michel Portal et al. 'An Overview of Non-Volatile Flip-Flops Based on Emerging Memory Technologies'. In: *Journal of Electronic Science and Technology* 12.2 (June 2014), pp. 173–181. DOI: 10.3969/j.issn.1674-862X.2014.02.007 (cited on page 108).

[107] Benjamin Ransford, Jacob Sorber, and Kevin Fu. 'Mementos: System Support for Long-Running Computation on RFID-Scale Devices'. In: *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XVI. Newport Beach, California, USA: Association for Computing Machinery, 2011, pp. 159–170. DOI: 10.1145/1950365.1950386 (cited on page 108).

[108] D. Balsamo et al. 'Hibernus: Sustaining Computation During Intermittent Supply for Energy-Harvesting Systems'. In: *IEEE Embedded Systems Letters* 7.1 (2015), pp. 15–18 (cited on page 108).

[109] P. A. Hager et al. 'A scan-chain based state retention methodology for IoT processors operating on intermittent energy'. In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. 2017, pp. 1171–1176 (cited on page 108).

[110] A. Krakovinsky et al. 'Impact of a laser pulse on HfO2-based RRAM cells reliability and integrity'. In: *2016 International Conference on Microelectronic Test Structures (ICMTS)*. 2016, pp. 152–156 (cited on page 108).

[111] A. Krakovinsky et al. 'Thermal laser attack and high temperature heating on HfO2-based OxRAM cells'. In: *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 2017, pp. 85–89 (cited on page 108).

[112] E. Valea et al. 'SI ECCS: SECure context saving for IoT devices'. In: *2018 13th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*. 2018, pp. 1–2 (cited on page 108).

[113] E. Valea et al. 'Providing Confidentiality and Integrity in Ultra Low Power IoT Devices'. In: *2019 14th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*. 2019, pp. 1–6 (cited on page 108).