Confiance matérielle : solutions de conception de verrouillage numérique

Quang-Linh Nguyen

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

En Systèmes Automatiques et Microélectroniques

École doctorale : Information, Structures, Systèmes

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

CONFIANCE MATÉRIELLE : SOLUTIONS DE CONCEPTION DE VERROUILLAGE NUMÉRIQUE
HARDWARE TRUST: DESIGN SOLUTIONS FOR LOGIC LOCKING

Présentée par Quang-Linh NGUYEN

Le 18/05/2022

Sous la direction de Bruno ROUZEYRE, Marie-Lise FLOTTES et Sophie DUPUIS

Devant le jury composé de

Roselyne Chotin, Maîtresse de Conférences, Sorbonne Université, CNRS, LIP6	Rapporteur
Alberto Bosio, Professeur d'université, École Centrale de Lyon, CNRS, INL	Rapporteur
Vincent Beroulle, Professeur d'université, LCIS, Univ. Grenoble Alpes, Grenoble INP	Président
Sophie Dupuis, Maîtresse de Conférences, LIRMM, Université de Montpellier, CNRS	Co-Encadrante
Marie-Lise FLottes, Chargée de recherche CNRS, LIRMM, Université de Montpellier, CNRS	Co-Encadrante
Bruno Rouzeyre, Professeur d'université, LIRMM, Université de Montpellier, CNRS	Directeur de thèse

UNIVERSITÉ DE MONTPELLIER

Abstract

In the globalized semiconductor supply chain, increasing outsourcing to external contractors, e.g., offshore foundry, results in possible exposure of hardware designs to adversaries. Consequently, security threats such as integrated circuit (IC) overproduction, intellectual property (IP) piracy and hardware Trojans have emerged, which poses a serious impact on the technology industry as well as the national security. Logic locking is a holistic Design-for-Trust method that can address those issues. By inserting key-controlled logic, logic locking allows locking the circuit functionality with a key only known by the IP/IC owner. The requirements for logic locking are to secure the logic locking key as well as effectively disrupting the circuit functionality. The security of logic locking is greatly challenged by oracle-guided attacks, notably the SAT attack. The attack is based on SAT solver and takes advantages of open scan chain access. Ensuring functionality disruption necessitates sufficient corruption at outputs, which is influenced by the insertion strategy of key-controlled logic. In this thesis, we develop secure and effective logic locking schemes, by focusing on three aspects, insertion strategy, SAT-secure logic lock and scan chain protection. For optimizing output corruption, we propose a key-gate insertion strategy called KIP. Based on signal probability analysis, the strategy selects nodes where corruption at such location is highly observable at multiple outputs. Furthermore, we propose a secure logic locking technique called SKG-Lock. It consist of inserting so-called switchable key-gates that are controlled by a switch controller. The technique provides provable SAT resilience and significant output corruption, especially when using KIP for insertion. Finally, we propose a scan controller for protecting logic locking against oracle-guided attacks. The scan controller introduces a key-based authentication mechanism, thus, prevents unauthorized access to the scan chains once the IC is deployed in the field.

* * *

Des menaces de sécurité telles que la surproduction de circuits intégrés (IC), le piratage de la propriété intellectuelle (IP) ou l'insertion de chevaux de Troie matériels sont apparues dues à l'externalisation de certaines phases de conception / production auprès de sous-traitants externes, par exemple une fonderie offshore. Le verrouillage logique ou Logic Locking est une méthode de conception permettant de contrecarrer de telles menaces. Il consiste à insérer une logique contrôlée par une clé connue uniquement du propriétaire du circuit et activée en retour de fabrication par celui-ci. L'utilisation de tout autre clef conduit à un dysfonctionnement du circuit. Toutefois, la sécurité apportée par le verrouillage logique a été fortement remise en cause par l'apparition d'attaques dites guidées par oracle, notamment l'attaque SAT. Celle-ci est basée sur l'utilisation d'un solveur SAT et tire parti de l'accès aux chaines de scan nécessaires à un test efficace du circuit. Garantir un dysfonctionnement suffisant pour que le circuit soit inemployable avec une mauvaise clef nécessite une corruption suffisante au niveau des sorties, qui est elle-même influencée par la stratégie d'insertion de la logique contrôlée par la clé. Dans cette thèse, nous développons des schémas de verrouillage logique sécurisés et efficaces, en nous concentrant sur trois aspects, la stratégie d'insertion, le verrouillage logique sécurisé SAT et la protection de la chaîne de scan. Pour optimiser la corruption de sortie, nous proposons une stratégie d'insertion de porte-clés appelée KIP. Sur la base d'une analyse de probabilité de signal, la stratégie sélectionne les nœuds où la corruption influence un grand nombre de sorties. De plus, nous proposons une technique de verrouillage logique sécurisée appelée SKG-Lock. Elle consiste à insérer des porte-clés dits commutables qui sont contrôlés par un contrôleur de commutations. Cette technique offre une résistance aux attaques SAT et une corruption de sortie importante, en particulier lorsque la stratégie KIP est utilisée. Enfin, nous proposons un contrôleur de chaines de scan pour protéger le verrouillage logique contre les attaques guidées par oracle. Le contrôleur de balayage implante un mécanisme d'authentification basé sur une clé, empêchant ainsi tout accès non autorisé aux chaînes de scan.

Résumé

Introduction

Le modèle économique de l'industrie des semi-conducteurs est actuellement majoritairement un modèle d'externalisation. Les coûts de fabrication étant devenus prohibitifs, l'externalisation du processus de fabrication vers des fonderies étrangères est devenue une tendance majeure. Ceci conduit à une exposition accrue de la propriété intellectuelle de la conception du matériel à des acteurs externes, potentiellement non fiables. En outre, les techniques de rétro-ingénierie matérielle sont devenues plus avancées, ce qui facilite la tâche des adversaires. En raison de la perte de contrôle sur l'utilisation de la propriété intellectuelle et de la multiplication des adversaires potentiels dans cette chaîne d'approvisionnement mondialisée, *la surproduction de circuits intégrés*, *le piratage de la propriété intellectuelle* et *l'insertion de chevaux de Troie matériels* sont devenus des sources majeures de préoccupation en matière de cyber sécurité [1]–[3]. Ces menaces ont un impact sérieux sur l'industrie des semi-conducteurs ainsi que sur la sécurité nationale, car des secteurs critiques tels que l'armée et l'aérospatiale reposent fortement sur l'électronique [4]–[6]. Par conséquent, des méthodes préventives, appelées *Design-for-Trust* (conception pour la confiance), sont d'une importance capitale pour regagner la confiance dans la conception des circuits intégrés.

De nombreuses approches récentes de *Design-for-Trust* introduisent des mécanismes préventifs au moment de la conception [7] : les techniques de prévention des chevaux de Troie matériels améliorent la détectabilité des chevaux de Troie, empêchent l'insertion de chevaux de Troie ou contournent les cœurs infectés par des chevaux de Troie [8] ; le *watermarking* et le *fingerprinting* démontrent la conformité de la propriété intellectuelle [9] ; le *metering* permet d'identifier les circuits intégrés fabriqués [10] ; le *camouflaging* empêche la rétro-ingénierie [11], [12] ; enfin, le verrouillage numérique, ou *Logic Locking*, permet de «verrouiller» la fonctionnalité d'un circuit avec une clé secrète, connue uniquement du propriétaire de la propriété intellectuelle [13]–[15].

Verrouillage Numérique

Le verrouillage numérique insère de la logique combinatoire dans un circuit, contrôlée par des entrées de clé supplémentaires. Il a plusieurs vertus par rapport aux autres solutions de conception pour la confiance. Il permet de

lutter contre le piratage de la propriété intellectuelle puisque le circuit extrait des outils de rétro-ingénierie n'est pas un circuit fonctionnel. Il fournit au concepteur un contrôle post-fabrication sur les circuits intégrés afin d'éviter la surproduction, puisque seul le bureau d'études ou un partenaire autorisé peut déverrouiller la fonctionnalité des circuits intégrés verrouillés. Parmi les solutions de Design-for-Trust, le verrouillage numérique offre la protection la plus polyvalente. Il suppose que seul le concepteur, c'est-à-dire le bureau d'études ou le propriétaire de la propriété intellectuelle, est digne de confiance. Par contre, tous les acteurs de toutes les étapes ultérieures dans le flot de production de circuits intégrés sont de potentiels adversaires. En résumé, le verrouillage numérique peut être déployé sur un cœur de propriété intellectuelle, pour prévenir de son utilisation non légale, ou sur un circuit en conception, pour empêcher la surproduction par une fonderie ; tout en empêchant la rétro-ingénierie, et l'insertion de chevaux de Troie.

Un moyen simple mais efficace de verrouiller la fonctionnalité d'un circuit consiste à insérer des «portes-clé», i.e., des portes logiques, par exemple XOR/XNOR, contrôlées par une entrée de clé. Le signal sur lequel une porte est insérée sera corrompu si une clé incorrecte est appliquée à son entrée. Ensuite, l'erreur se propagera et provoquera potentiellement de la corruption aux sorties du circuit. Avec une clé incorrecte, une corruption de sortie significative est souhaitable de sorte que le circuit verrouillé soit inutilisable. La stratégie d'insertion de la porte a donc son importance, puisqu'elle a un impact direct sur la corruption de la sortie.

Une conception à verrouillage numérique contient également une unité de gestion des clés et une structure de conception pour le test (*Design for Test* - DfT). L'unité de gestion des clés, qui comprend généralement une mémoire inviolable, est chargée de stocker et de délivrer la clé aux entrées de clé du circuit verrouillé. La structure DfT, qui comprend généralement comprend des chaînes de scan (registres à décalage), vise à faciliter les tests post-silicium.

Dans le flux de conception d'un circuit intégré, le verrouillage logique peut être appliqué au niveau netlist, i.e. sur la description du circuit en interconnexions de portes logiques, qui est obtenue après l'étape de synthèse logique. La netlist résultante, appelée *netlist verrouillée*, peut ensuite être transformée en description physique – layout - par synthèse physique. Pour la fabrication du circuit intégré, la description physique verrouillée peut être envoyée à des installations offshore, pour la fabrication, le test et la mise en puce – packaging. La fonderie fabrique les puces de silicium à partir de la vue physique verrouillée. Par conséquent, les tests sont effectués avant l'activation des circuits intégrés, c'est-à-dire sur les circuits intégrés verrouillés. Cela est possible car le test de fabrication, qui est essentiellement un test structurel, peut être effectué indépendamment de la fonctionnalité du circuit. Après la fabrication, le test et le packaging, les circuits intégrés sont déverrouillés avant d'être disponibles à l'achat. L'activation des circuits intégrés consiste à programmer la clé de verrouillage logique dans la mémoire inviolable du circuit

intégré. Cette opération peut être effectuée en interne lorsque les puces sont renvoyées au bureau d'études, ou à distance chez un tiers de confiance.

Les vulnérabilités du verrouillage numérique

Le modèle de menace pour le verrouillage logique suppose que les attaquants de la chaîne d'approvisionnement peuvent obtenir la netlist du circuit verrouillé et/ou un oracle. Une fonderie non fiable peut en effet effectuer une rétro-ingénierie du de la vue physique verrouillée pour obtenir la netlist verrouillée. Un attaquant peut également obtenir la netlist verrouillée en négociant avec la fonderie ou en procédant à la rétro-ingénierie d'un circuit intégré. Les circuits intégrés prêts à l'emploi, c'est-à-dire les circuits intégrés non verrouillés, notamment pour l'électronique grand public, peuvent être achetés sur le marché et accessibles à tous. Un circuit intégré non verrouillé fournit les paires d'entrée-sortie correctes et peut donc être utilisé comme *oracle* lors d'une attaque. Le scénario d'attaque dans lequel l'attaquant possède la netlist verrouillée et un circuit intégré déverrouillé est appelé *attaque guidée par un oracle*. D'autre part, les produits électroniques fabriqués sur mesure pour certains secteurs critiques tels que l'armée ou l'aérospatiale peuvent ne pas être disponibles à la vente pour les consommateurs. Le scénario d'attaque dans lequel l'attaquant possède uniquement la netlist verrouillée est appelé *attaque sans oracle*.

Le tournant de la recherche sur le verrouillage numérique a été l'introduction de l'attaque dite *SAT* [16], qui présente la technique d'attaque guidée par un oracle la plus puissante, capable de casser toutes les défenses précédentes. À chaque itération, cette attaque trouve un vecteur d'entrée distinctif (*Differential Input Pattern* - DIP) qui donne des sorties différentes pour deux valeurs de clés différentes. La principale force de l'attaque SAT repose sur deux facteurs importants :

(i) la capacité d'éliminer une grande quantité de clés erronées pour chaque DIP,

(ii) la possibilité de cibler n'importe quelle partie combinatoire du circuit grâce à l'accès aux chaînes de scan.

La première génération de techniques de verrouillage numérique résistantes à l'attaque SAT [17], [18] vise à réduire le nombre de fausses clés pouvant être éliminées par chaque DIP. Cependant, ce résultat est obtenu grâce à un compromis sur la corruption des sorties.

Contributions

La sécurité du verrouillage numérique est mise en péril par les attaques guidées par un oracle, notamment les attaques basées sur des solveurs SAT. De plus, un verrouillage numérique efficace doit fournir une corruption des

sorties suffisante. Dans cette thèse, nous avons pour objectif de proposer des méthodes de verrouillage numérique à la fois *sûres* et *efficaces*. Les contributions de cette thèse sont au nombre de trois :

(i) KIP: une stratégie d'insertion de portes-clé optimisée pour la corruption des sorties, basée sur l'analyse de l'impact des portes-clé sur la probabilité des sorties du circuit ;

(ii) SKG-Lock: une technique de verrouillage numérique efficace et sécurisée contre les attaques SAT qui permet de contrôler la commutation des portes-clé ;

(iii) Scan controller: une solution de chaîne de scan sécurisée pour la prévention des attaques guidées par un oracle, qui empêche l'accès non autorisé aux chaînes de scan.

État de l'art

Depuis 2008, la recherche sur le verrouillage numérique s'est développée de manière prolifique, suivant un jeu du chat et de la souris entre les défenses et les attaques. Comme l'introduction de « l'attaque SAT » a eu un impact durable sur la recherche sur le verrouillage numérique, la chronologie peut être considérée possédant deux ères : l'ère pré-SAT et l'ère post-SAT.

Dans l'ère pré-SAT, les techniques de verrouillage numériques étaient basées sur l'insertion de portes-clé. La corruption de la sortie était l'objectif le plus important de ces techniques. La corruption des sorties peut être caractérisée par les métriques suivantes : le *taux de corruption des sorties* présente la probabilité d'observer un ou plusieurs bits erronés au niveau du vecteur de sorties d'un circuit verrouillé ; la *couverture de la corruption des sorties* présente l'ampleur de la corruption propagée aux sorties du circuit i.e. le nombre de bits de sorties erronés cumulés; la *corruptibilité des sorties* présente la probabilité d'observer un ou plusieurs bits erronés au niveau du vecteur de sortie d'un circuit verrouillé. Le verrouillage numérique dit «basé sur les fautes» (*Fault based Logic Locking* - FLL) [19], [20] fournit la stratégie d'insertion de porte-clés la plus efficace en termes de corruption de sorties. Pour insérer chaque porte-clé, cette méthode trouve le signal ayant le plus fort impact sur les fautes, une métrique basée sur la simulation de fautes.

L'attaque SAT a été capable de briser toutes les techniques basées sur les portes-clé lorsqu'elle a été introduite pour la première fois. L'attaque est basée sur un solveur SAT (satisfaisabilité booléenne). À chaque itération, il trouve un DIP, qui est ensuite appliqué à l'oracle. La réponse correcte obtenue est ajoutée à la formule du solveur en tant que nouvelle contrainte, ce qui permet d'éliminer une partie de l'espace de recherche des clés pour l'itération suivante. Progressivement, jusqu'à ce que plus aucun DIP ne puisse

être trouvé, l'attaque élimine des valeurs de clé jusqu'à déduire la clé correcte. Les circuits séquentiels ne sont sensibles à cette attaque que lorsque *les chaînes de scan sont accessibles* à l'attaquant.

Dans l'ère post-SAT, la résilience aux attaques SAT devient une nécessité pour le verrouillage logique. Deux directions vers la résilience aux attaques et la corruption de sortie pour le schéma de verrouillage logique ont été développées. La première direction est d'augmenter drastiquement le temps de calcul de l'attaque. Le verrouillage crypté [21], [22] empêche l'attaque SAT en augmentant de manière exponentielle le temps d'exécution de chaque itération mais est très coûteux et attaquable par d'autres types d'attaques. Le verrouillage numérique basé sur une fonction «1-point» [17], [18] a été la première contre-mesure contre l'attaque SAT dont la sécurité est prouvée. Il consiste à insérer un verrou avec une fonction «1-point» dans le circuit. Pour chaque valeur d'entrée, ce verrou n'altère les sorties du circuit que pour une valeur de clé. Par conséquent, toutes les valeurs d'entrée sont des DIP ne pouvant éliminer qu'une valeur de clé unique, que l'attaque SAT doit donc parcourir exhaustivement pour exclure toutes les valeurs de clé incorrectes. Cependant, pour ce type de technique, la corruption de la sortie est fortement compromise. Les techniques les plus récentes de verrouillage logique, dans la même mouvance que les travaux développés dans cette thèse, commencent permettre de créer des protections qui résistent aux attaques SAT, tout en augmentant la corruptibilité [23]–[26]. Une autre direction est celle des défenses basées sur la protection des chaînes de scan, qui peuvent empêcher non seulement l'attaque SAT mais aussi d'autres attaques guidées par oracle. Les techniques existantes comprennent le verrouillage des chaînes de scan [27], [28], le blocage des chaînes de scan [29]–[31] et le contrôle des chaînes de scan [32], [33].

Du côté offensif, des attaques ont également été proposées pour évaluer la sécurité des défenses récentes. Grâce à son efficacité, l'attaque SAT a été la base de diverses attaques suivantes. Les attaques SAT guidées par oracle [34]–[37] peuvent inclure des étapes ou des contraintes supplémentaires, ou utiliser un solveur plus polyvalent. Les attaques sans oracle visent à supprimer la logique de protection insérée pour récupérer la netlist originale [38], [39] ou à analyser la structure du circuit pour deviner la valeur de la clé [40], [41].

Stratégie d'insertion de portes-clé pour un verrouillage numérique efficace

Cette contribution est une stratégie évolutive d'insertion de porte-clés basée sur l'analyse de probabilité (KIP) qui est optimisée pour les métriques de corruption des sorties.

Le principe de la stratégie proposée est de classer chaque signal d'un circuit en fonction d'une métrique appelée *score de corruption de sortie*. Pour chaque signal, cette métrique reflète l'impact sur les sorties si le signal est corrompu à cause d'une porte-clé insérée. Comme pour FLL, cette stratégie émule la corruption en insérant des fautes de collage à ce signal. L'impact d'une faute donnée sur les sorties est mesuré en enregistrant la différence de probabilité des sorties de valoir 1/0 avec et sans la faute. Par conséquent, le score de corruption des sorties est obtenu comme le produit de la différence de probabilité totale et du nombre de sorties impactées. L'insertion de portes-clé sur les signaux ayant un score élevé aura un impact sur la plupart des sorties pour la plupart des valeurs d'entrée, ce qui entraîne une couverture de corruption de sortie et un taux de corruption élevés.

L'algorithme de KIP comprend deux étapes, le classement des signaux et la sélection des signaux sur lesquels insérer une porte-clé. Le score de corruption de sortie de chaque signal est calculé et les signaux sont classés en fonction de leur score dans un ordre décroissant. Ensuite, la sélection des signaux commence par le signal ayant le score le plus élevé. Ici, un critère supplémentaire est appliqué : un seul signal est choisi parmi les signaux ayant le même score, afin d'éviter les séries de portes-clé. Le temps d'exécution de la stratégie KIP est essentiellement l'étape de classement des signaux. Par rapport à FLL, KIP ne classe les nœuds qu'une seule fois, alors que FLL refait le classement à chaque fois qu'une porte-clé est insérée.

Effectué avec l'insertion d'une porte-clé XOR, KIP obtient des résultats optimaux dans toutes les métriques de corruption de sortie ; un taux de corruption de sortie de 100%, une couverture de corruption de sortie de 100% et une corruptibilité de sortie de 50% sont obtenus dans plusieurs circuits. KIP est plus évolutif que FLL en termes de temps d'exécution ; par exemple, avec le circuit c7552, la stratégie KIP s'est terminée en 10 minutes, alors que l'algorithme FLL prend 2 heures pour le même circuit.

Un verrouillage numérique sécurisé et efficace grâce à des portes-clé commutables

Cette seconde contribution est une nouvelle technique de verrouillage numérique sécurisé, appelée *SKG-Lock*, qui vise à contrecarrer les attaques basées sur un solveur SAT, tout en maintenant une corruption de sortie significative.

Les deux composants fondamentaux de SKG-Lock sont des portes-clé commutables (*Switchable key-gates* - SKGs) et un contrôleur de commutation (*Switch controller* - SWC). Les portes-clé commutables possèdent trois entrées, deux signaux de commande — une entrée de la clé K_A et un signal de commutation *sw* — et un signal S provenant du circuit verrouillé. Le contrôleur de commutation contrôle les signaux de commutation des portes-clé commutables. Une conception générale pour un contrôleur de commutation est

un comparateur entre l'entrée clé K_D et les entrées du circuit, construit avec une rangée de portes XNOR et une cascade de portes AND. Cette structure du contrôleur de commutation produit des signaux de commutation, chacun provenant de chaque nœud de la cascade AND, c'est-à-dire que chacun a une corruptibilité différente.

Deux ensembles d'entrées de clé, la clé d'activation (*Activation key* - K_A) et la clé de leurre (*Decoy key* - K_D) sont introduits : K_A est connectée aux portes-clé commutables ; K_D est connectée au contrôleur de commutation. Elles proviennent toutes deux d'une mémoire inviolable. Bien que le circuit soit déverrouillé en insérant la bonne valeur de clé K_A pour chaque SKG, la valeur de clé K_D est importante pour obtenir la résilience contre les attaques SAT.

La structure proposée de SKG-Lock consiste à insérer plusieurs portes-clé commutables contrôlées par différents signaux de commutation, chacun ayant une corruptibilité différente. Le signal de commutation ayant la corruptibilité la plus faible est la sortie d'une fonction 1-point entre K_D et les entrées primaires, qui génère une complexité maximale pour l'attaque SAT. L'utilisation d'autres signaux de commutation avec des corruptibilités plus élevées permet d'augmenter la corruption des sorties de l'ensemble du système. La stratégie KIP est utilisée pour l'insertion des portes-clé commutables afin d'optimiser davantage la corruption des sorties. L'utilisation de plusieurs signaux de commutation et de plusieurs portes-clé commutables crée des connexions multiples, c'est-à-dire une intrication structurelle, entre le contrôleur de commutation et le circuit verrouillé.

Nous proposons l'analyse la sécurité de SKG-Lock contre diverses attaques. La résilience maximale contre l'attaque SAT est obtenue tant qu'il y a au moins une porte-clé commutable contrôlée par le signal de commutation de moindre corruptibilité, indépendamment des autres portes-clé commutables insérées avec une corruptibilité plus élevée. L'utilisation de plusieurs portes-clé commutables avec des corruptibilités différentes empêche d'autres attaques basées sur des solveurs SAT, comme l'attaque approximée AppSAT [34] et l'attaque Bypass [35]. Des contre-mesures contre des attaques potentielles sans oracle [38], [41] contre SKG-Lock peuvent être incorporées avec des modifications structurelles : des portes-clé supplémentaires pour « cacher » le contrôleur de commutation, ce qui permet de ne pas avoir des signaux « reconnaissables » avec une très faible probabilité de commutation ; des portes-clé commutables modifiées, déverrouillables avec un bit de K_A et un bit de K_D. Notez que ces modifications structurelles n'ont pas d'impact sur la résilience de SKG-Lock contre les attaques SAT.

Un verrou à fonction 1-point peut être recréé avec un contrôleur de commutation et une seule porte-clé commutable contrôlée par le signal de commutation de moindre corruptibilité. Cette option «légère» de SKG-Lock atteint le même niveau de résilience SAT tout en nécessitant un surcout en aire deux

fois moindre par rapport aux verrouillages numériques à fonctions 1-point connus [17], [18].

Comparé aux techniques résilientes contre les attaques SAT [17], [18], [24], [26], SKG-Lock fournit une corruption de sortie significativement plus élevée et une meilleure résilience contre les attaques. L'évaluation du surcout en aire montre que SKG-Lock a un surcout acceptable sur de petits circuits. Notez que ce surcout est proportionnel à la taille des clés, et indépendant de celle du circuit à verrouiller. Donc le surcout est moindre pour de plus gros circuits.

Contrôleur de chaîne de scan pour protéger le verrouillage numérique contre les attaques guidées par un oracle

Cette contribution est un contrôleur de chaîne de scan qui limite l'accès aux chaînes de scan aux seuls utilisateurs autorisés. L'idée principale de la solution proposée est d'utiliser une authentification basée sur une clé (*Scan Access Key* – K_S) pour contrôler les opérations de décalage servant au contrôle/«remplissage» (*shift-in*) et à l'observation (*shift-out*) des chaînes de scan. Le contrôleur de chaîne de scan contient un registre à décalage à rétroaction linéaire (*linear feedback shift register* - LFSR) de n bits et un comparateur. La valeur de KS, stockée dans une mémoire inviolable, est la «graine» du LFSR. Le contrôleur de chaîne de scan "verrouille" le signal d'activation de la chaîne de scan (*scan enable* - SE) en le mettant à "0". SE n'est mis à "1" que lors de l'insertion continue d'une séquence correcte via la broche supplémentaire K_T de la clé de test. Le flux de bits K_T correct doit correspondre à la sortie du LFSR, qui dépend de K_S.

Dans le flot de conception, le contrôleur de chaîne de scan est inséré pendant l'insertion du DfT. Pendant le test de fabrication, pour permettre l'accès aux chaînes de scan sans partager la valeur secrète K_S, une valeur codée en dur est utilisée comme valeur temporaire de K_S pendant le test et cette valeur peut être partagée avec le testeur. Une broche supplémentaire utilisée pour permettre cette option est ensuite supprimée après le processus de packaging. Ensuite, K_S, ainsi que la clé de verrouillage numérique, sont programmées dans la mémoire inviolable pendant la phase d'activation du circuit intégré. Pour le débogage et l'analyse des défaillances, un testeur autorisé qui obtient K_S et la structure du LFSR peut construire un modèle équivalent du LFSR pour générer le flux de bits K_T requis qui active les chaînes de scan.

Le schéma de verrouillage numérique proposé, composé d'une protection à deux couches basée sur le contrôleur de chaînes de scan et la technique d'insertion de portes-clé utilisant la stratégie KIP, est très efficace et sécurisé contre de nombreuses attaques.

Le contrôleur de chaînes de scan proposé bloque les opérations de décalage dans les chaînes de scan effectuées par les attaquants, ce qui empêche la mise en œuvre de puissantes attaques guidées par oracle qui nécessitent un accès aux chaînes de scan [16], [34], [36]. Par exemple, l'attaque SAT nécessite de contrôler et d'observer respectivement les entrées et les sorties de la partie combinatoire attaquée. Dans ce cas, l'attaquant ne peut appliquer aucun DIP généré aux entrées de la partie combinatoire, et il ne peut pas non plus lire la réponse correcte pour l'élimination de la mauvaise clé. Par conséquent, l'attaque ne peut pas être mise en œuvre.

En adaptant le modèle de menace du verrouillage numérique, nous avons identifié des schémas d'attaque potentiels sur le contrôleur de chaînes de scan. Grâce au mécanisme basé sur un comparateur, les attaques qui visent à deviner la clé d'accès aux chaînes de scan ne peuvent pas être plus efficaces que la force brute. La suppression et le contournement potentiels du contrôleur de chaînes de scan peuvent être détectés. La solution est testable, facile à intégrer et supporte des tests complets. En outre, elle présente un faible surcoût par rapport aux défenses basées sur les chaînes de scan.

Conclusion

Les exigences du verrouillage numérique sont de sécuriser la clé de verrouillage et de perturber efficacement la fonctionnalité du circuit verrouillé. Pour assurer la perturbation de la fonctionnalité, il faut une corruption suffisante des sorties, qui est influencée par la stratégie d'insertion de la logique contrôlée par la clé. Pour évaluer la sécurité du verrouillage numérique, de nombreuses attaques ont été développées. L'attaque SAT a introduit la méthode d'attaque la plus efficace, basée sur la satisfiabilité booléenne et guidée par un oracle. L'utilisation d'un solveur SAT permet de distinguer rapidement la bonne clé de celles qui ne le sont pas, de sorte que chaque itération du processus d'attaque peut éliminer un grand nombre de clés erronées. En outre, l'attaque tire parti des chaînes de scan dans l'oracle pour cibler directement chaque partie combinatoire verrouillée, ce qui rend l'attaque faisable en un temps restreint. Par conséquent, une forte résistance contre l'attaque SAT a été une priorité pour les techniques de verrouillage numériques sécurisées. Les techniques existantes basées sur les fonctions 1-point sont manifestement protégées contre l'attaque SAT, mais au prix d'une corruption minimale des sorties. De plus, non seulement l'attaque SAT, mais d'autres attaques guidées par un oracle, exploitent également les chaînes de scan de l'oracle. Dans cette thèse, nous identifions trois aspects pour développer un verrouillage numérique sûr et efficace, à savoir la stratégie d'insertion des portes-clé, le verrouillage numérique sécurisé contre les attaques SAT et la protection des chaînes de scan. Finalement, les contributions de cette thèse construisent deux schémas de verrouillage numérique :

(i) **SKG-Lock + KIP** : ce schéma consiste en l'utilisation de la protection SKG-Lock basée sur la stratégie KIP pour l'insertion des portes-clé commutables. Il s'agit d'une technique de verrouillage numérique générique qui peut être utilisée dans tous les cas par n'importe quel défenseur. Elle fournit une sécurité prouvable contre les attaques SAT ainsi qu'une grande résilience contre d'autres attaques efficaces guidées par un oracle ou sans oracle. Cependant, l'utilisation d'une structure de fonction 1-point peut laisser une vulnérabilité structurelle. Ce schéma montre des résultats significatifs à la fois dans le taux de corruption des sorties et la couverture de corruption. Le surcoût en surface est raisonnable.

(ii) **KIP + Scan Controller** : ce schéma consiste en une protection à deux couches, l'insertion de portes-clé XOR avec la stratégie KIP pour le verrouillage du circuit et un contrôleur de chaînes de scan pour la protection des chaînes de scan. Il convient dans les cas où les manipulations dans l'insertion DfT et la génération de test sont autorisées. Une protection complète contre les attaques guidées par oracle est réalisée. Cependant, le schéma peut encore être vulnérable aux attaques potentielles sans oracle. La technique des portes-clés utilisant la stratégie KIP fournit une corruption de sortie très élevée. Le surcoût des portes-clé insérées dans le circuit protégé est faible. Le contrôleur de chaînes de scan ajoute un surcoût matériel acceptable à l'infrastructure DfT du circuit. Il entraîne également une surcharge dans le temps de test.

Chaque schéma présente une direction différente vers un verrouillage logique sûr et efficace, avec des avantages différents. Nous avons également identifié les points clés à améliorer dans les solutions proposées.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Sophie Dupuis, Marie-Lise Flottes and Bruno Rouzeyre. I am grateful for their guidance, support and patience. Thank you for this opportunity to work on an exciting and challenging project.

I would like to extend my thanks to Roselyne Chotin and Alberto Bosio for agreeing to review my work and being member of my thesis committee, and to Vincent Beroulle for agreeing to join the thesis committee.

I deeply treasure the time I had working at LIRMM. It was a privilege to work side by side with such brilliant minds. My thanks also go to the staffs of LIRMM for all the administrative support.

I would like to thank the collaborators from MOOSIC project and SAFEST project. It was a pleasure to have such in-depth discussions with you.

I extend my thanks to all my friends in France and in Vietnam, who were with me and encouraged me so much throughout this journey.

I would like to express my heartfelt gratitude to my family back home, my parents and my younger brother, for their unwavering support and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

ATPG	Automatic Test Pattern Generation
CAC	Corrupt-And-Correct
CNF	Conjunctive Normal Form
DfT	Design-for-Test
DIP	Distinguish Input Pattern
DTL	Diversified Tree Logic
EDA	Electronic Design Automation
FF	Flip-Flop
FI	Fault Impact
FLL	Fault-based Logic Locking
FSM	Finite State Machine
HD	Hamming Distance
HT	Hardware Trojan
IC	Integrated Circuit
IP	Intellectual Property
JTAG	Joint Test Action Group
LFSR	Linear Feedback Shift Register
MUX	Multiplexer
OSAT	Outsourced Semiconductor Assembly and Test
PUF	Physical Unclonable Function
RE	Reverse Engineering
RLL	Random Logic Locking
s-a-f	stuck-at-fault
SAT	Boolean Satisfiability
SC	Secure Cell
SE	Scan Enable
SFLL	Stripped Functionality Logic Locking
SKG	Switchable Key Gate
SMT	Satisfiability Modulo Theory
SoC	System-on-Chip
SWC	Switch Controller
TM	Test Mode

Chapter 1

Introduction

Contents

1.1 The Demand for Design-for-Trust

1.1.1 Hardware Security Threats Emerged from Semiconductor Supply Chain

The outsourcing business model currently dominates the semiconductor industry. Ever-shrinking technologies have rapidly raised the cost of manufacturing integrated circuits (ICs). Currently, constructing a fabrication plant with advanced technologies (5nm to 3nm) costs more than $10 billions [42]. Therefore, outsourcing the fabrication process to offshore foundries has become a major trend [43]. This leads to the rising of *fabless semiconductor companies*, which focus on IC design without manufacturing capacity. Fabless companies range from small startups to large corporations, e.g, Qualcomm, Broadcom, Nvidia and Apple. Besides, modern System-on-Chip (SoC) design is getting more complex, integrating multiple modules. To reduce design effort and time-to-market, semiconductor companies rely on reusable hardware intellectual properties (IPs), provided by *IP vendors*. Hardware design IPs range from hard IPs (e.g., GDSII layout files), firm IPs (e.g., synthesized netlists) to soft IPs (e.g., synthesizable HDL codes). The market for semiconductor IP is growing rapidly with leading IP vendors such as ARM, Rambus and Xilinx. For IC design companies, hardware design is indeed valuable intellectual property. However, industrial practice has paid little attention into protecting hardware IPs.

The general IC production flow is illustrated in Fig. 1.1. A fabless company, so-called *design house*, brings up the IC design from the idea/specification to the physical design/layout[1]. During the design process, the design house may use hardware IPs from *IP vendors*. The layout design is sent to a *foundry*, which is often located offshore and belongs to an external company. Here the layout is implemented onto silicon. Consequently, fabricated silicon may be sent to an outsourced semiconductor assembly and test (OSAT) company for testing and packaging. Finally finished ICs are released into the market and purchased by *users*.

In fact, the globalized supply chain leads to exposure of hardware design IP to external and possibly unreliable actors. Given that the industry promotes a tough competition and laws enforcing IP protection are different in each country, trust violation among entities in the supply chain is very likely. Research community has identified and investigated possible security threats on hardware coming from rogue entities in the supply chain such as *IC overproduction*, *IP piracy* and *Hardware Trojan insertion* [1]–[3]. The number of recorded cases on hardware attacks and violations has been alarmingly increasing [4]–[6], [44]. These threats pose a serious impact on the technology

[1]The layout is in GDSII format, the de facto industry standard for data exchange of IC layout.

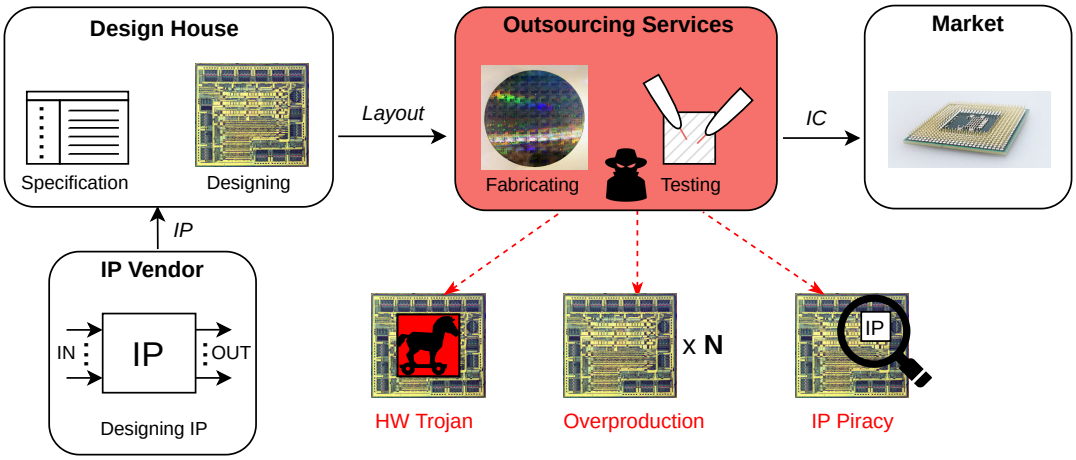


FIGURE 1.1: Semiconductor supply chain and emerging security threats.

industry as well as the national security since critical sectors such as military and aerospace rely heavily on electronics.

With the access to the layout design as well as the technology library, i.e., the detail of each cell, an offshore foundry is the most likely attacker. Possible threat scenarios where the foundry is the adversary are as follows.

- IC overproduction: An untrusted foundry may copy the IC design and build more than the required number of ICs that the design house orders, and then sell the excess ICs in the gray market for extra profit.
- IP piracy: With the help of reverse engineering tools, the foundry can extract valuable IPs, which belong to IP vendors or the design house, from the layout of the IC design. They can then exploit the recovered IP or reuse it for their own profit.
- Hardware Trojan: Hardware Trojan (HT) [8] is a malicious modification or addition to the specified design of an IC. Circuits infected by HTs may show undesired behaviors like changes in functionality, leakage of information, performance degradation or even denial of service. An ill-natured foundry could reverse-engineer the design and insert HTs by modifying the lithographic masks. One can notice that the reliance on commercial design tools and reusable hardware IPs also leads to a risk of purchasing from untrusted vendors: IPs already contain HTs; tools implant HTs into the design.

Reverse engineering (RE), a legalized work yet the enabling tool for mentioned hardware threats, has become more accessible [45] and more advanced [46]–[48]. This allows any attacker with only access to the fabricated chip to obtain the layout design by depackaging, delayering, and imaging the chip [49]. The attacker can further recover the IC design at any desired abstraction level with reverse engineering: from GDSII/layout to gate-level netlist [46],

and from gate-level netlist to register transfer level (RTL) or behavior level [47], [48], [50].

IC overproduction, IP piracy and Hardware Trojan have become alarming due to the lack of control for IC designer/IP owner and the increasingly advanced adversaries. Therefore, preventive methods, referred to as Design-for-Trust, are of vital importance to regain trust in IC design.

1.1.2 Design-for-Trust solutions

Numerous recent *Design-for-Trust* approaches introduce preventive mechanisms at design time [7]: hardware Trojan prevention techniques increase HT detectability, prevent HT insertion or bypass HT-infected cores; watermarking and fingerprinting demonstrate compliance with the IP ownership; hardware metering provides identification for fabricated ICs; IC camouflaging prevents reverse engineering; last but not least, logic locking allows locking the circuit functionality with a key only known by the IP holder.

Hardware Trojan Prevention

A *hardware Trojan* generally contains two main components, a trigger and a payload (cf. Fig. 1.2). The HT remains dormant until triggered. Once its trigger has detected an expected event or condition, the payload is activated to perform a malicious action.

With HT prevention methods, designers can revise a defense strategy during design-time that aids test-time HT detection as well as ensures secure runtime operations [8]. These techniques harden circuit design at different abstraction levels. Methods applied at layout level can hinder HT insertion by replacing "dummy" filler cells with testable cells [51]. Hence, the substitution of those cells by HTs would be detectable at test time. At gate level, designers can deploy techniques that aim to manipulate the transition probability of signals during testing and expose stealthy HTs implanted at low-activity signals. These techniques are based on adding dummy scan flip-flops [52] or obfuscating the netlist with key-controlled gates [53] (cf. Fig. 1.2), which is referred to as logic locking (cf. details in a following subsection). Techniques at system level and RTL provide structures to ensure trustworthy operations even with the presence of HT-infected components; these techniques are based on hardware redundancy [54] or security wrappers for monitoring IP cores [55].

Watermarking & Fingerprinting

Watermarking and *fingerprinting* [9] are the first Design-for-Trust methods dedicated to IP protection. They consist in inserting the signatures of IP owner (watermark) and legal IP user (fingerprint) onto the circuit design. This is achieved with additional constrains during the design procedure. Although

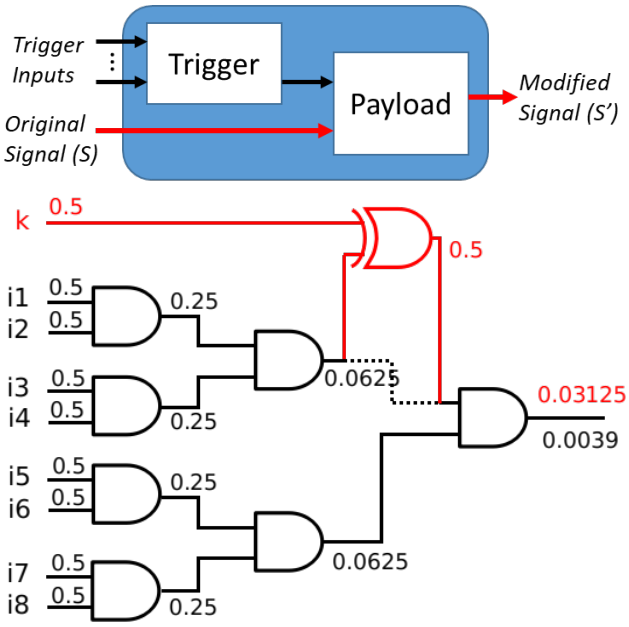


FIGURE 1.2: (a) Hardware Trojan structure. (b) Obfuscating the netlist with XOR gates to increase HT detectability.

the functionality of the IP is kept intact, these constraints create distinct structures and properties in the implementation of the IP. These additional features will be used as a proof of IP ownership or IP user identification on the fabricated IC; hence, they are required to be detectable in a remote and non-destructive way.

Hardware Metering

A contract foundry may illegally overproduce chips more than the number from the design house's order. However, watermarking and fingerprinting cannot prevent overproduction; the overbuilt ICs still carry the authentic watermark and fingerprint implemented by the design house. Therefore, *hardware metering* [10] was introduced as a protocol to allow the design house to have post-fabrication control over their ICs. Its principle is to embed a unique tag to each IC and ensure that the tag is under the control of the design house. Early proposals of hardware metering were based on passive approach using a physical unclonable function (PUF) or digitally stored serial numbers for IC identification.

Camouflaging

Reverse engineering allows better understanding of the structure and functionality of IC, but also facilitates adversaries in conducting hardware attacks. *IC camouflaging* [11], [12] was proposed to prevent RE by obfuscating the layout design. This method designs structures or cells, e.g., NAND, NOR and XOR, with difference functionalities that look alike from the top view

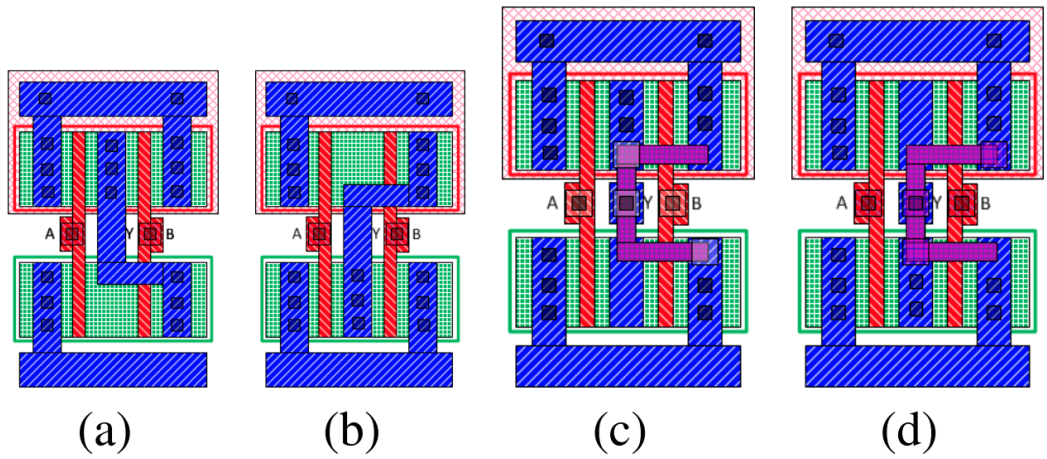


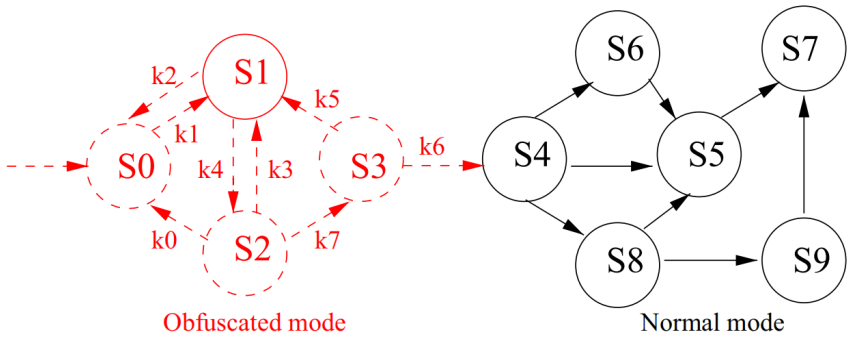
FIGURE 1.3: The standard cell layouts of NAND gate (a) and NOR gate (b), and their camouflaged cell layouts (c)(d) [11].

of nanoscale imaging. Camouflaged cells are inserted throughout the netlist with a given insertion strategy. An example of camouflaged cells is illustrated in Fig. 1.3. While the standard layouts of 2-input NAND and NOR gates have different metal routing and are recognizable by visual inspection, their camouflaged counterparts have identical metal layers. Camouflaging is considered effective against physical IC reverse engineering since the technology knowledge for RE is normally 2-3 generations behind the latest design technology; thus the dummy and true contacts between certain adjacent metal layers might be indistinguishable for the RE tools.

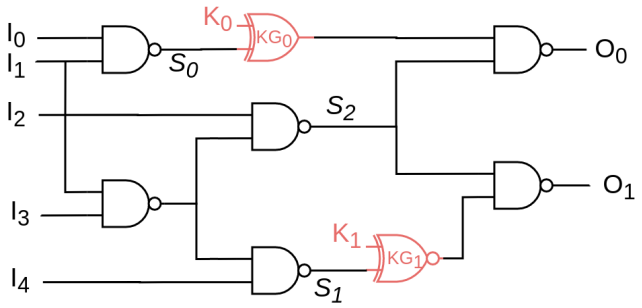
Logic Locking

Logic locking[2] [13]–[15] consists in modifying the circuit structure with additional logic gates and/or flip-flops so that a key is required for the circuit to function correctly. Logic locking can be categorized into combinational logic locking [18], [24], [26], [56], [57] and finite state machine (FSM) locking [58]–[61]; while the former approach inserts additional circuitry into the combinational parts of the design, the latter inserts into the sequential parts. In *FSM locking*, with additional state elements, the original FSM, termed normal mode, is expanded with a new set of states forming a so-called obfuscated mode (cf. Fig. 1.4). Upon power-up, the locked circuit is in the obfuscated mode; however, it must be set in the normal mode to behave correctly. Thus, a key, in this case an input sequence, is required to bring the circuit from the obfuscated to the normal mode. *Combinational logic locking* embeds additional gates controlled by dedicated key-inputs into the design. The locked design behaves as the original one only upon the application of the correct key value at the key-inputs; otherwise, it outputs erroneous values. In both techniques,

[2]Logic locking is also referred to as logic encryption, logic obfuscation and logic masking in the literature.



(a) FSM locking ($KeySequence = k_1k_4k_7k_6$)



(b) Combinational logic locking ($K_0K_1 = 01$)

FIGURE 1.4: Examples of FSM and combinational logic locking.

the design provides the correct functionality only if the provided key value is correct. Only the IP owner/design house who applies logic locking to the design knows the secret key value. Therefore, in following stages of the IC production flow, without knowledge about the key, untrusted entities are hindered from maliciously using the logic-locked IP. After production, the circuits are unlocked, by the design house or a trusted partner, before being released into the market.

As logic locking techniques are proposed for digital ICs, the same principle can be used to protect analog and mixed-signal ICs. Techniques proposed in [62], [63] lock the digital parts of analog and mixed-signal circuits. Any disturbance in the function of digital parts potentially sensitizes the analog parts. Hence, without the correct key, the circuit exhibits a performance degradation.

The Advantages of Logic Locking

Logic locking exhibits several virtues over other Design-for-Trust solutions. Dedicated HT prevention techniques cannot protect ICs against other threats. Although hardware watermarking, fingerprinting and metering techniques can hamper IP piracy by means of ownership proof and identification, they do not actively prevent overusing IPs, overbuilding ICs or increasing the complexity of RE. Logic locking and camouflaging both counter IP piracy

since the extracted design from RE tools is not a working design. However, only logic locking provides the designer with post-fabrication control over ICs to prevent overproduction[3]. This is because only the design house or an authorized partner can unlock the functionality of logic-locked ICs; whereas camouflaging does not involve an explicit secret key. Furthermore, camouflaging cannot hinder an untrusted foundry because the foundry requires information about camouflaged cells for the correct fabrication. Therefore, among Design-for-Trust solutions, logic locking offers the most versatile protection. It assumes only the designer, i.e., the design house or the IP vendor, to be trusted and assumes actors in any subsequent stage in the IC production flow to be the adversary. Logic locking can be used at IP core level to restrict illegal IP reuse and prevent IP reverse engineering, and at system level to prevent IC overproduction and, to some extent, prevent hardware Trojan insertion.

Logic locking, especially the combinational method, has received tremendous attention from the research community and the industry. The industry has become more aware of threats from semiconductor supply chain [64]–[67]. To facilitate the adoption of security measure into IC design flow, electronic design automation (EDA) companies have introduced logic locking into their automation platforms: Mentor proposed TrustChain solution [68]; Synopsys has collaborated with academic partners to integrate logic locking in their tool suite [69], [70]. The number of publications about logic locking has seen a great surge for the last decade [13]. In addition, international academic competitions dedicated to logic locking help raise awareness about the topic among students and researchers [71], [72].

In this thesis, we focus specifically on combinational logic locking. In the rest of the thesis, we will use the term logic locking to refer to combinational logic locking.

1.2 Logic Locking

1.2.1 Architectural View of Logic Locking

How to Lock a Circuit

The modified IC design flow including logic locking is depicted in Fig. 1.5. Logic locking can be applied on gate-level netlist, which is obtained after the logic synthesis stage. It modifies the netlist by adding key-controlled logic. The result netlist, referred to as *locked netlist*, can then be transformed into layout by physical synthesis.

A simple yet effective way to lock circuit functionality is to insert key-gates, as proposed in EPIC [56], the first logic locking technique. A *key-gate* is a

[3]Logic locking is considered as an active hardware metering technique.

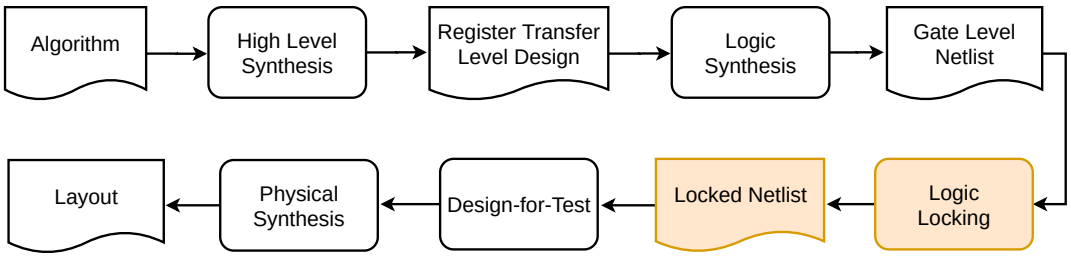


FIGURE 1.5: Design flow with logic locking.

logic gate controlled by a key-input (cf. Fig. 1.4). Key-gates can be in the form of XOR/XNOR gates [19]–[21], [56], [57], multiplexers (MUX) [20], [73], or AND/OR gates [53], [74].

The signal at which a key-gate is inserted will be corrupted if an incorrect key is applied at its key-input. Thus, the error is propagated and potentially causes corruption at circuit outputs, also referred to as *output corruption*. With an incorrect key, significant output corruption is desirable so that the locked circuit is useless. The insertion strategy for key-gates indeed has an impact on output corruption. The EPIC solution introduced random insertion of XOR key-gates; however, the strategy is not optimized for output corruption. Consequently, fault-based logic locking (FLL) [19] presents a more optimized strategy that finds key-gate locations based on fault simulation so that it maximizes the number of affected outputs and the number of input patterns that lead to erroneous outputs.

Building Blocks of a Logic-Locked IC

Along with the locked design/netlist, a logic-locked circuit contains other building blocks including a key management unit and Design-for-Test (DfT) structure [75], as illustrated in Fig. 1.6.

The key management unit is in charge of storing and delivering the key to the key-inputs of the locked netlist. The main component is a tamper-proof memory [76], which is used to store security assets such as the logic locking key. It can be built from non-volatile memories or one-time programmable memories. The key is configured into this memory during IC activation stage. Given the implementation of logic locking as mentioned previously, all fabricated ICs will have a common logic locking key. In case where the IC activation is performed remotely by an untrusted party or user, it is desirable that each fabricated device has a different key value. A PUF, combined with a crypto engine, can form a public-key infrastructure that enables secure remote IC activation protocol [56], [77]. Based on process variations, a PUF [78] generates a unique challenge-response pair for each fabricated device. Therefore, it can be used to customize a unique key for each chip [19].

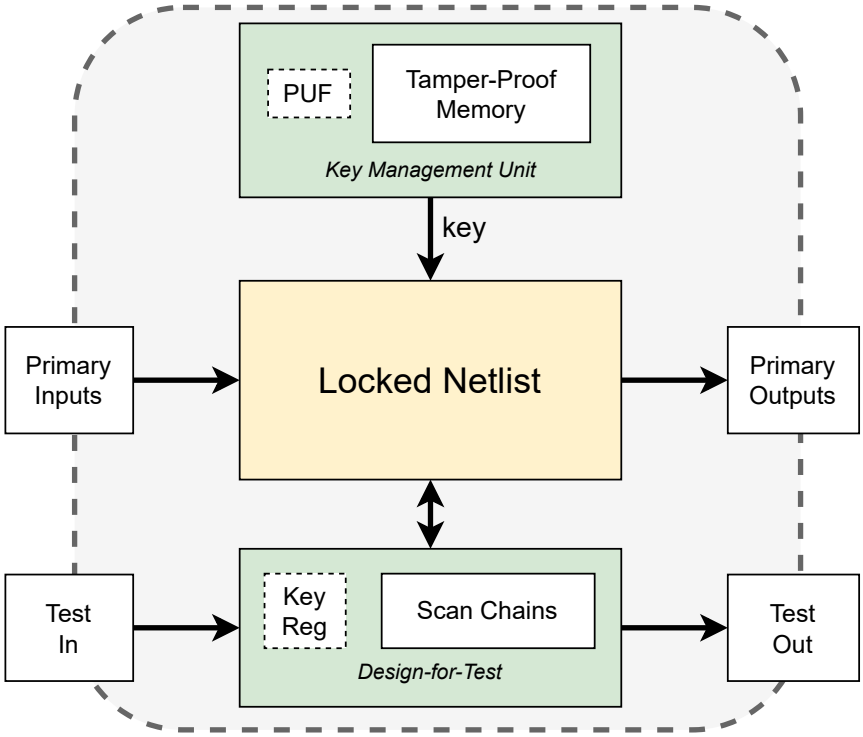


FIGURE 1.6: A logic-locked IC contains the locked netlist, a key management unit and Design-for-Test structure.

Design-for-Test structure is an important part of modern IC to ensure the quality of fabricated chip. Scan design [79] is a fundamental DfT approach where flip-flops (FFs) in the netlist are transformed into scan FFs and connected in series to form *scan chains*. Scan chains facilitate structural test by allowing control and observation of internal nodes of the netlist. The key from the tamper-proof memory may be fed to the key-gates through dedicated registers, which are termed as key-registers. Key-registers can be used to enable structural test before IC activation [29], [30].

Each component has implication on the security of the logic locking scheme. In the scope of this thesis, we focus on powerful attacks on logic locking that exploit the locked netlist and the scan chains.

1.2.2 Vulnerabilities of Logic Locking

Threat Model

Fig. 1.7 depicts the supply chain adopting a logic locking scheme. The green background indicates the stages performed in-house or by a trusted partner; the red background indicates potential adversaries, which involve in all outsourced stages. With logic locking, outsourced stages including fabricating, testing and packaging can be done on locked design/IC by offshore foundry

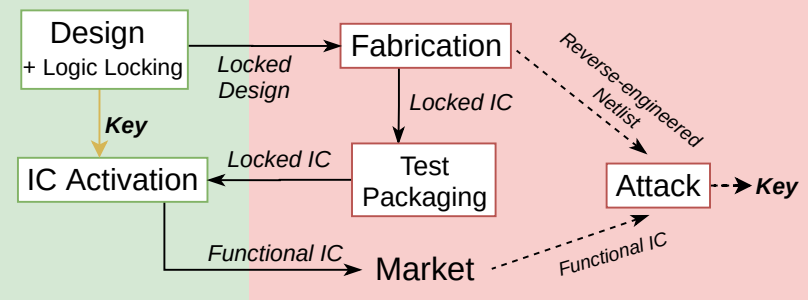


FIGURE 1.7: IC supply chain and threat model on logic locking.

and OSAT. The foundry fabricates silicon chips of the locked design. Consequently, testing is performed before IC activation, i.e., on locked ICs [80], [81]. This is possible since the manufacturing test, which is essentially structural test, can be performed irrespective of circuit functionality. After fabrication, testing and packaging, the ICs are unlocked before being available for purchase. IC activation consists in programming the logic locking key into the tamper-proof memory in the IC. This can be done in-house when chips are shipped back to the design house or remotely at a trusted third-party.

The foundry, the OSAT company and the end user are potential attackers who want to recover the logic locking key. The attacker's capabilities are defined as follows. The foundry can reverse-engineer the locked design layout to obtain the locked netlist. The end user or OSAT can obtain the locked netlist by trading with the foundry or by reverse-engineering an IC. Ready-to-use ICs, i.e., unlocked ICs, especially for consumer electronics, can be purchased from the market and accessible to anyone. An unlocked IC provides the correct input-output pairs, hence, can be used as the *oracle* in the attack. The attack scenario where the attacker owns the locked netlist and an unlocked IC is termed as *oracle-guided attacks*. On the other hand, electronics custom-made for certain critical sectors such as military or aerospace may not available for purchase by any consumer. The attack scenario where the attacker owns the locked netlist is termed as *oracle-less attacks*.

The locked netlist and the oracle are powerful assets for realizing attacks on logic locking. The netlist can be used for simulation and generation of input patterns that may reveal information about the key. The oracle allows the attacker to obverse circuit function not only through the primary inputs/outputs but also through the test interface, i.e., scan chains, that provides access into the FFs of the circuit.

The Most Important Attack: the SAT Attack

The turning point for research in logic locking was the introduction of the *SAT attack* [16], presenting a powerful attack technique that is able to break all precedent defenses. The attack has ultimately steered the succeeding logic

locking research: state-of-the-art logic locking solutions aim to be resilient against this attack; more recent attacks are derivatives of the SAT attack.

The SAT attack is an oracle-guided attack, based on Boolean Satisfiability (SAT) solver. It works as an iterative process. In each iteration, the attack finds a so-called Distinguish Input Pattern (DIP) that results in different outputs for two different keys. The key search space is reduced iteratively until no more DIP can be identified. At that point, only the correct key remains in the search space. With naive logic locking techniques, the attack can rule out all incorrect keys within a few iterations, irrespective of the input size and the key size; whereas the brute force approach would require exponential time with respect to the number of inputs and key-inputs. The SAT attack can be applied directly on combinational netlists. Sequential circuits are susceptible to this attack only when the *scan chains are accessible* to the attacker.

Ultimately, the main strength of the SAT attack relies on two important factors:

(i) the capability to eliminate a large quantity of wrong keys with each DIP,

(ii) the capability to target any combinational part of the circuit thanks to the access to the scan chains.

The first generation of SAT-resilient logic locking techniques [17], [18] aims to reduce the number of wrong keys that can be eliminated by each DIP. However, this is achieved due to a trade-off in output corruption.

The access to scan chains of the oracle provides the possibility to mount the SAT attack on sequential circuits, as well as plethora of other oracle-guided attacks [82], [83]. Therefore, a complementary defense for the scan chains is of interest to prevent a wide range of such attacks on logic locking. Nevertheless, a scan-based defense may require modification in DfT insertion, test generation and test procedure [84]. This approach is suitable for defenders such as a design house who is in charge of the design and the production test of their IC products.

1.3 Thesis Contributions

The security of logic locking is jeopardized by oracle-guided attacks, notably SAT-based attacks. Furthermore, effective logic locking needs to provide sufficient output corruption. In this thesis, we aim to propose logic locking schemes that are *secure* and *effective*. The contributions of this thesis are as follows.

First, we propose a new insertion strategy, called *KIP*, that aims to maximize output corruption. It searches for locations where corruption is easy propagated to outputs and affects the most outputs possible. It ranks each node in

the netlist according to how easy the corruption is easy propagated to outputs and how many outputs it affects. The score for each node is measured by its impact on the probability of circuit outputs, using transitional probability analysis. For evaluation, we locked benchmark circuits by insertion XOR key-gates according to the proposed strategy. Compared to the FLL strategy, the KIP strategy is more scalable, provides more precise ranking and overcome some drawbacks of FLL.

Next, we tackle the question: Can a logic locking technique limit the pruning capability of each DIP in SAT-based attacks without compromising the output corruption? To this end, we propose a novel logic locking technique, namely *SKG-Lock*, based on so-called switchable key-gates (SKGs) and additional decoy key-inputs. We theoretically validate that SKG-Lock achieves maximum resilience against the SAT attack, regardless of the number of switchable key-gates and their corruptibility. Moreover, SKGs can be inserted using the proposed KIP strategy to maximize output corruption. By taking advantage of multiple SKGs, the proposed SKG-Lock provides tremendously higher output corruption, better structural entanglement and better resilience against SAT-based attacks while incurring reasonable overhead, compared to related SAT-resilient techniques. Countermeasures for SKG-Lock against potential oracle-less attacks are presented.

Powerful oracle-guided attacks on logic locking rely on the scan chain access. A defender can implement a scan-based defense to ensure that the scan chains are inaccessible to the attacker; however this requires alterations to DfT insertion and production test generation. To this end, we propose a scan protection for logic locking, based on a *scan controller*. The scan controller introduces a key-based authentication mechanism, thus, prevents unauthorized users from enabling the scan chains once the IC is deployed in the field. A logic locking scheme, including the scan controller and a simple logic locking technique such as XOR key-gate insertion using the KIP strategy, is secure against powerful attacks and has low overhead compared to related scan-protected schemes. Security analysis shows the scan controller's ability to secure the scan access key and its robustness against tampering. Full testing is supported when performed by authorized partners without leaking secret data to adversaries. In addition, the solution is scalable and easy to integrate.

The remainder of the thesis is organized as follows:

- Chapter 2 provides a detailed background on logic locking, including state-of-the-art logic locking schemes and attacks.
- Chapter 3 introduces KIP, a new key-gate insertion strategy based on analyzing the impact of key-gates on the probability of circuit outputs.
- Chapter 4 proposes SKG-Lock, a secure logic locking scheme that allow controlling of the switching of key-gates.

- Chapter 5 presents a secure scan solution, comprised of a scan controller to prevent unauthorized scan access, for logic locked circuits.
- Chapter 6 concludes the thesis and gives insights on the future perspectives on logic locking.

Chapter 2

Background & State-of-the-Art

Contents

2.1 Brief History of Logic Locking

Fig. 2.1 depicts the timeline of research on logic locking. Since 2008, research on logic locking, especially combinational logic locking[1], has prolifically developed, following a cat-and-mouse game between defenses and attacks.

As the introduction of the SAT attack has a lasting impact on logic locking research, the timeline can be seen as pre-SAT and post-SAT era. In pre-SAT era, logic locking techniques were based on key-gate insertion. Output corruption is the most important objective for such techniques. However, the SAT attack presented an unprecedented attack method that is able to break every key-gate based techniques at that time. In post-SAT era, SAT-attack resilience becomes a must for logic locking. Point-function based logic locking was the first SAT-attack countermeasure that is provably secure. However, for this type of technique, output corruption is greatly compromised. Since 2017, two directions towards attack resilience and output corruption for logic locking have been developed. Recent secure logic locking techniques are based on novel structures that are resistant against the SAT attack while increasing corruptibility. Another direction is scan-based defenses that can prevent not only SAT attack but other oracle-guided attacks. Along with defenses, attacks have been proposed to evaluate the security of their targeting defenses. Recent attacks on logic locking includes SAT-based oracle-guided attacks and oracle-less attacks.

2.2 Key-Gate based Logic Locking

Key-gate based logic locking refers to techniques that involve insertion of key-gates. EPIC [56] was the first technique to introduce a key-gate insertion method. It is also known as random logic locking (RLL) as the key-gates are inserted randomly. Succeeding solutions have introduced more strategic insertion, taking into account output corruption as the most important criterion. As different attacks reveal vulnerabilities of key-gate based techniques, different insertion criteria have been introduced to mitigate such vulnerabilities.

[1] Compared to sequential logic locking, combinational logic locking is more scalable and easier to integrate. Moreover, various frameworks have been proposed to characterize combinational logic locking.

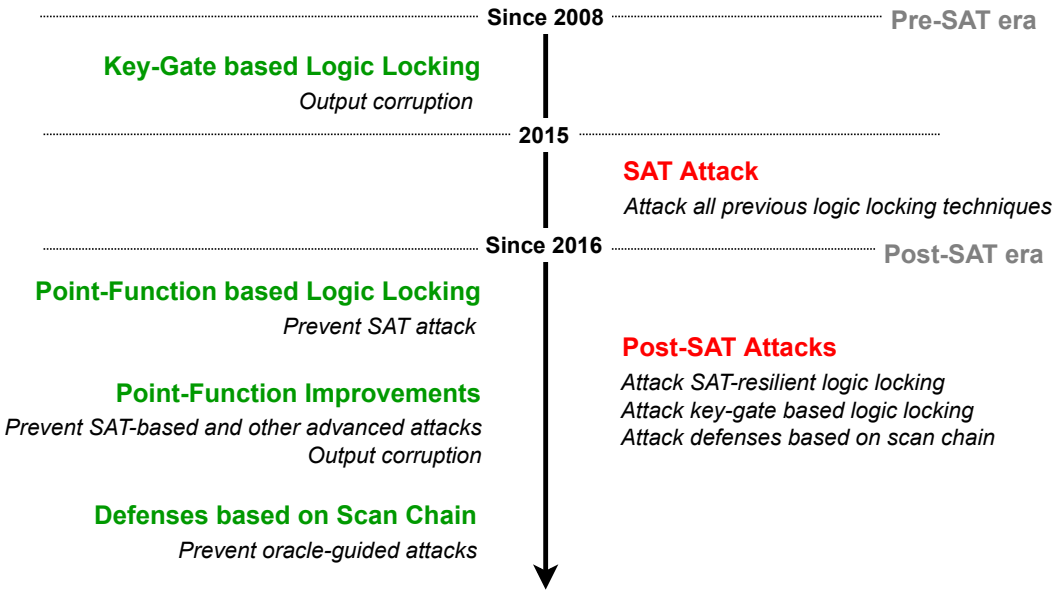


FIGURE 2.1: Timeline of research on logic locking.

2.2.1 Output Corruption

Output corruption indicates the effectiveness of logic locking in making the circuit behave incorrectly when applied with an incorrect key. Output corruption can be estimated with the following metrics.

Metrics for Output Corruption

For estimating output corruption of a logic locking scheme on a circuit, one can apply to the locked circuit N_K key values, each with N_I input patterns and observe its output O_L containing m bits. The same input patterns are applied to the original circuit to observe its output O. The difference, measured by Hamming distance (HD), between the output of the locked circuit and the original one is recorded to compute the following metrics.

Definition 1 (Output corruption rate) *Output corruption rate*[2] *presents the probability of observing erroneous bit(s) at the output vector of a locked circuit. It is measured by the percentage of input patterns that lead to errors at circuit outputs when any incorrect key value is applied:*

$$\frac{1}{N_K \times N_I} \sum_{i=1}^{N_K} \sum_{j=1}^{N_I} c \times 100\% \tag{2.1}$$

Where,

[2]It is also termed output error rate or error rate [24], [34].

$$c = \begin{cases} 1, & \text{if } HD(O_L(I_j, K_i), O(I_j)) \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

Note that, even if there is corruption for a large number of input patterns, but the number of outputs corrupted is minimal, it may not be sufficient for preventing the usage of locked circuits. For instance, in the image processing domain, even if the least significant bit is always wrong (100% output corruption rate), the circuit can still be used.

Definition 2 (Output corruption coverage) *Output corruption coverage presents the magnitude of corruption propagated to circuit outputs. It is characterized by the maximum number of output bits that can be corrupted, i.e., the maximum Hamming distance between the outputs on applying any wrong key and the correct key:*

$$\frac{l}{m} \times 100\% \tag{2.3}$$

Where,

$$l = max(HD(O_L(I_j, K_i), O(I_j)))\forall i \in [1..N_K], j \in [1..N_I] \tag{2.4}$$

Definition 3 (Output corruptibility) *Output corruptibility*[3] *presents the probability of corruption at any circuit output. It is estimated by the average Hamming distance between the outputs on applying any wrong key and the correct key [85]:*

$$\frac{1}{N_K \times N_I \times m} \sum_{i=1}^{N_K} \sum_{j=1}^{N_I} HD(O_L(I_j, K_i), O(I_j)) \times 100\% \tag{2.5}$$

For output corruption rate and output corruption coverage, the higher, the better; whereas, for output corruptibility, 50% is the optimum value so that the circuit mimics random logic and hence this maximizes obscureness for attackers using brute-force approach [19], [20], [86].

Different logic locking techniques/architectures provide different degrees of output corruption. FLL can achieve optimal output corruptibility, whereas SAT-resilient techniques based on point-function, which will be discussed in Section 2.3.2, only have extremely low percentage of output corruption rate.

2.2.2 Insertion Strategies for Output Corruption

Significant output corruption is desirable so that a locked circuit without the correct key is useless. A key-gate supplied with a wrong key bit corrupts the signal at its location and disturbs signals in its fanout. However, corruption

[3] It is often termed Hamming distance [20] due to how it is computed.

at one node can impact more outputs than corruption at another node in the netlist. Hence, the strategy for selecting key-gate locations can be revised to maximize output corruption.

Output corruption can be seen as equivalent to fault propagation. To find key-gate locations that result in high output corruptibility, FLL [19], [20] introduced a new metric called the fault impact (FI). For each node of the netlist, it measures the impact of a stuck-at-0 (sa0) fault and a stuck-at-1 (sa1) fault at this node on the circuit's behavior. The FI value of a node is calculated with simulation by the number of patterns that detect the stuck-at-0/1 fault at this node (N_{N_I-0}/N_{N_I-1}) and the total number of affected output bits (N_{O-0}/N_{O-1}):

$$(N_{N_I-0} \times N_{O-0}) + (N_{N_I-1} \times N_{O-1}) \tag{2.6}$$

The fault impact of each node in the netlist is calculated by simulating the circuit with a number of random input patterns. A key-gate is inserted at the node with highest fault impact. This process is repeated in the modified circuit to insert each following key-gate and until the required number of inserted key-gates is achieved. This selective insertion allows significant output corruption, up to 100% output corruption rate and 50% output corruptibility. However, this strategy is computation-intensive: it relies intensively on simulation; fault impact of every node in the circuit is recalculated for the insertion of each key-gate.

The simulated fault impact metric of FLL is also used in succeeding corruption-focus techniques. Weighted Logic Locking (WLL) was proposed in [87], where each key-gate is controlled by two or more key bits to increase its actuation probability (a 2-input XOR key-gate has the actuation probability of 50%). Using such key-gates can achieve optimal output corruptibility with less key-gates, compared to using 2-input XOR key-gates. Fault simulation is used to find locations for key-gates. A technique proposed in [88] combines different criteria to find optimal locations for key-gates. It first find the overlap region of several logic cones; thus, corruption at any node in this region can impact several outputs. Then it assesses the fault impact of nodes in such regions and the interdependence between nodes to select locations for key-gate insertion.

The fault impact metric is simulation-intensive and, hence, not scalable. The work in [89] proposed to use centrality indicator to select key-gate locations. The netlist is represented as a graph where each gate is a vertex and each wire is an edge in the graph. Centrality indicator is used to find the most significant vertices in the graph by measuring metrics such as closeness-centrality and between-centrality. Such significant nodes are in the fanout of several inputs and the fanin of several outputs. It takes less time to insert key-gates than FLL; however, its output corruption is less optimized. This presents a

trade-off between computation time of insertion strategy and output corruption.

2.2.3 Attacks & Countermeasures

The key sensitization attack [57], also the first oracle-guided attack, aims to leak the key by sensitizing each of the key bits to a primary output. An automatic test pattern generation (ATPG) tool is used on the locked netlist to generate test patterns that allow sensitizing a key bit without being masked or corrupted by other key bits. In naive insertion strategies such as RLL, there is rarely interference between each key bit's path to the outputs; hence, the attack can target key bits individually. The test patterns are applied to the oracle through its test interface to observe the key bits from the received test responses. In order to thwart this attack, Strong Logic Locking (SLL) [21], [57] is proposed to prevent the sensitization of key bits on an individual basis. The proposed idea is to find key-gate locations with complex interference among them. This strategy maximizes the number of key-gates that are pairwise-secure, i.e., sensitizing one key bit requires the control of the other key bit and vice versa.

Attacks on logic locking also deploy the divide-and-conquer approach [90], [91]. The locked netlist can be divided into logic cones, each corresponds to each circuit output and is composed of gates in its transitive fan-in. With vulnerable insertion strategies, for each protected logic cone, i.e., the logic cone that contains key-gates, the number of key inputs may be relatively small. The attacker can target logic cones individually from the cone with the smallest number of key-inputs. Brute force [91] or differential power analysis (DPA) [90] can be used to resolve the protected cone. A defense strategy in [88] suggests to find a region in the netlist that is the overlap of several logic cones and only select key-gate locations in this region. Thus, each protected logic cone contains all the inserted key-gates.

Oracle-less attacks have also been proposed to counter key-gate based logic locking. The attack model assumes that the attacker has access to a locked netlist obtained from reverse engineering but no access to an activated IC. The attackers can distinguish key-inputs from primary inputs and identify key-gates in the netlist. In addition, they are aware of the insertion strategy.

The Redundancy attack [40] is based on logic redundancy identification. The realization behind the attack is that the locked netlist presents more logic redundancy if embedded with an incorrect key value compared to when embedded with the correct key value. Modern synthesis tools are able to remove redundancies, which are associated with untestable faults. Since logic locking is applied on a synthesized netlist, the insertion of extra key gates for logic locking does not affect the fault coverage of the netlist when the correct

activation key is applied; whereas, an incorrect key may affect the testability due to the introduction of redundancies. An ATPG tool can be used to detect the presence of untestable faults caused by logic redundancy, hence deducing the correctness of the applied key. The attack deduces the value for each key-bit separately. The same authors also proposed a insertion strategy to counter the attack [92]. The proposed strategy find locations where a key-gate inserted does not alter the redundancy level of the netlist.

The SAIL attack is an oracle-less attack based on machine learning [93]. It requires the locked netlist and the locking algorithm. Naive insertion of key-gates may leak the key due to the inference between the key-gate type and its key bit. Hence, a following resynthesis step involves bubble pushing, i.e., inserting inverters and transmitting them through gates nearby. The main observation is that the structural transformation due to resynthesis is local and predictable. SAIL leverages machine learning to revert these structural changes to their pre-synthesis state. Knowing the locking strategy, the attacker can build a training data set containing pre-synthesis and post-synthesis topology. The truly random logic locking (TRLL) technique [30] introduces an insertion technique to prevent this attack. Besides inserting XOR key-gates, it also finds locations of inverter gates in the netlist and replaces them with XOR key-gates. Thus, the correct key for a XOR key-gate is 0 in the former case and 1 in the latter case, hence, breaking the relation between the key-gate type and the key bit without the need for resynthesis.

In general, for each of these attacks, additional constraints to the insertion strategy are introduced as a countermeasure. A combination of these proposals can be revised to increase the security of logic locking scheme. Nevertheless, these attacks only exploit the structure of the locked circuit, but not the logical behavior compared to the SAT attack.

2.3 SAT Attack & Point-Function based Logic Locking

2.3.1 SAT Attack

The SAT attack [16] is an oracle-guided attack where the attacker has the access to two fundamental assets: (i) the locked netlist, (ii) an oracle with accessible scan chains.

The SAT solver is the processing kernel for this attack. The solver used in this proposal is based on the conflict-driven clause learning algorithm. It first assigns a value to a selected variable. As some assignments implicate conflicts, the formula is augmented with additional clauses to eliminate them. Then the solver backtracks based on the conflicts. This process is repeated until a solution is found or the problem is found to be unsatisfiable.

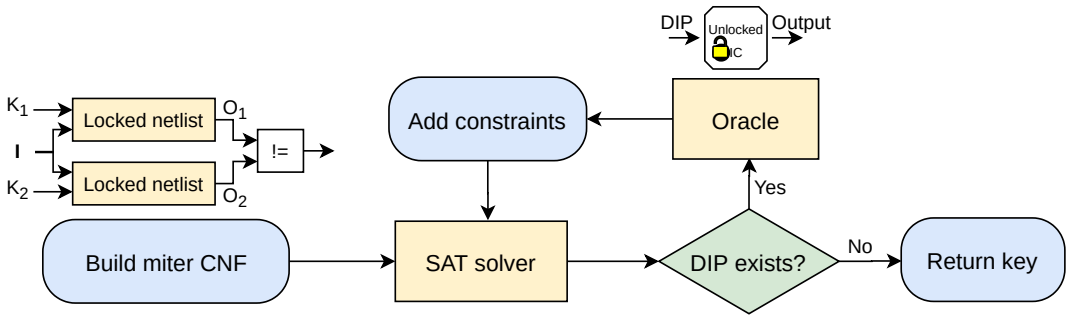


FIGURE 2.2: The procedure of the SAT attack.

The input for SAT solver is a formula in conjunctive normal form (CNF). A circuit netlist can be translated into CNF via the Tseitin transformation. For SAT attack, the input is from a miter circuit. This miter circuit contains two copies of the locked netlist, whose primary inputs are the same but key-inputs are different. Their outputs are compared to make the output of the miter circuit, e.g., 0 if their outputs are equal and 1 otherwise.

The SAT attack is an iterative process, as illustrated in Fig. 2.2. In each iteration, the SAT solver finds an assignment that results in output 0 for the miter circuit. This assignment contains a DIP and two key values. The DIP is then applied to the oracle to observe the golden output. This obtained input-output pair is added to the formula as constraints for the next iteration. The additional constraints could eliminate a larger portion of the key search space due to possible overlapping functionality of keys at a given input. This process is repeated until the problem of obtaining output 0 for the miter circuit is unsatisfiable, i.e., no more DIP can be found. This means that there are only correct key value(s) left in the key search space. Finally, the SAT solver deduces the correct key as a key value that respects all added constraints.

Key-gate based logic locking is highly vulnerable to the SAT attack. In such techniques, there are a large number of key values that result in wrong outputs for each input pattern; an example is illustrated in Fig. 2.3. Thus, each chosen DIP can rule out several wrong key values. In the example, the SAT attack is able to return the correct key after eliminating all incorrect keys values only with three iterations.

The resilience, i.e., the computation effort for a successful attack, against the SAT attack can be characterized by the execution time for a successful attack. The execution time of the SAT attack can be calculated as:

$$T = \sum_{n=1}^{N} t_n \tag{2.7}$$

where N is the number of iterations and t_n is the runtime for the nth iteration.

Increasing the resilience against the SAT attack can be achieved by

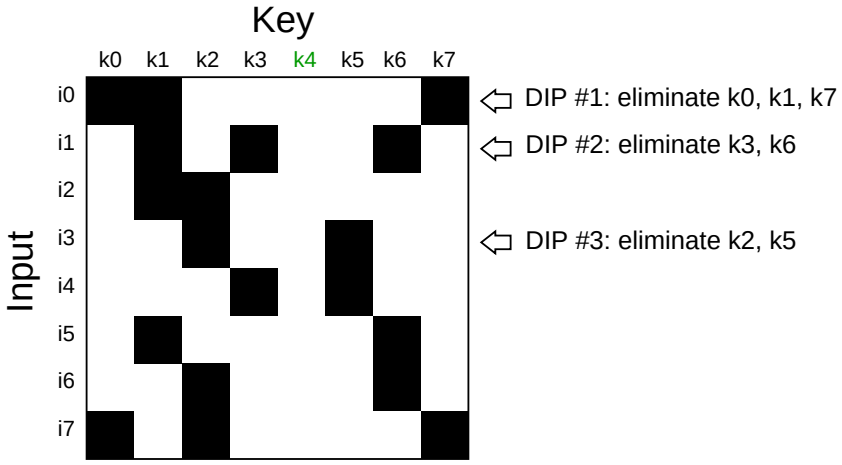


FIGURE 2.3: Example of SAT attack. Black boxes in the truth table represent the cases where outputs are corrupted.

- Increasing the number of DIPs,
- Increasing the time for each iteration.

2.3.2 Point-Function based Logic Locking

The most notable countermeasure against SAT attack is point-function based logic locking such as SARLock [18] and Anti-SAT [17], [94]. These techniques aim to make SAT attack exhaustive by rendering the number of DIPs exponential in the key size.

The structure of a point-function based technique, equivalent to SARLock [18], is depicted in Fig. 2.4. Such technique consists in inserting a key-controlled *point-function lock*, typically at an output of the circuit. The point-function lock contains two comparators connected in parallel, one for comparing primary inputs and key-inputs and another for comparing key-inputs and the hard-coded secret embedded with inverters. The inputs of an n-bit point-function lock are n-bit primary inputs and n-bit key-inputs. Since the point function is a Boolean function that outputs the value 1 for exactly one input pattern, the point-function lock only corrupts the circuit output(s) for one corresponding key value per input pattern, as seen in the truth table in Fig. 2.4. Therefore, with the SAT attack, each chosen DIP can only eliminate one wrong key value. To rule out all incorrect key values, the attack needs to select all input patterns as DIP for each iteration. Thus, returning the correct key requires $2^n - 1$ iterations.

Anti-SAT [17], [94] introduces a different structure of point-function lock. n-bit Anti-SAT block (cf. Fig. 2.5(a)) contains two sub-blocks with complementary functions (e.g., AND and NAND); each of them is controlled by n-bit key-inputs (hence, $2n$ key bits in total) and the same n-bit primary inputs. For each key value, the block outputs logic 1 for one primary input pattern.

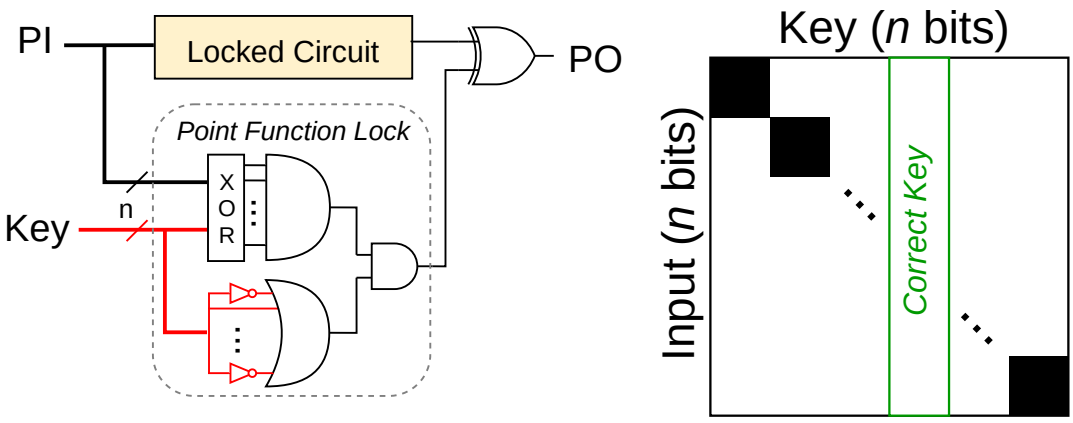
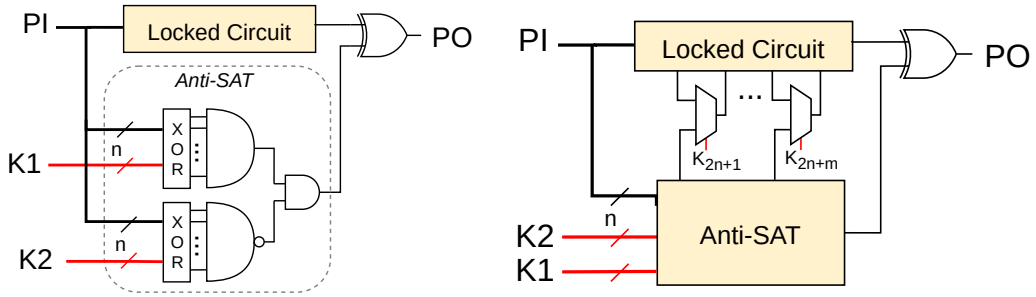


FIGURE 2.4: The structure and the truth table of a point-function lock.



(a) Anti-SAT block (b) Structural obfuscation for Anti-SAT

FIGURE 2.5: The structure of Anti-SAT.

Thus, the SAT attack has to run for $2^n - 1$ iterations to recover a correct key. A correct key value is the one where i key bit of K_1 vector equals to i key bit of K_2 vector. Thus, there are 2^n correct key values among 2^{2n} values.

In terms of structure, due to having only one connection with the locked circuit, the Anti-SAT block is isolated and separable. Structural obfuscation was proposed for Anti-SAT to create *structural entanglement* with the locked circuit. It consists in inserting additional key-controlled MUXes that connect internal wires of these two structures, as depicted in Fig. 2.5(b).

Although point-function based logic locking minimizes the pruning capability of each DIP, it suffers from low corruption. The output corruption rate of this scheme is $1/2^n$. Due to a low number of interconnections with the locked circuit, its output corruption coverage is inherently low. An effort to ensure both security and output corruption is to use a point-function based technique in conjunction with a key-gate based technique, called compound locking [17], [18]. However, this scheme is vulnerable to various approximate attacks [34], [95] (cf. Section 2.5.1).

For point-function based techniques, the protection logic block has only one

output, which makes it separable from the locked circuit. Moreover, the probability of its output is highly skewed (towards logic 1 or 0). Such structural vulnerabilities can be exploited by removal attacks [38], [96] (as detailed in Section 2.5.2)

Metric for SAT Resilience of Point-Function based Techniques

The SAT resilience of point-function based techniques is characterized by the number of SAT iterations for a successful attack.

Definition 4 (SAT resilience level) *SAT resilience level of a logic locking technique is n-secure if the SAT attack requires* 2^n *iterations to successfully unlock its locked circuit [24].*

SARLock and Anti-SAT achieve n-secure against SAT attack with n-bit point-function lock.

2.4 Post-SAT Logic Locking Techniques

2.4.1 Point-Function Lock Improvements

Recent logic locking techniques leverage the principle of point function to make SAT attack exhaustive. They introduce novel protection logic structures that can enable higher output corruption and mitigate structural vulnerability compared to conventional point-function lock.

Diversified Tree Logic (DTL) method [97] provides tunable output corruption with the modification of a few gates in the point-function based block. Such modification can be applied to the structures used in SARLock and Anti-SAT (cf. Fig. 2.4 and Fig. 2.5(a)), such as by replacing some of the gates in the AND tree with OR/AND/XOR gates, which increases the corruption rate. With this method, an increase in output corruption lead to a trade-off in SAT resilience, i.e., the number of SAT iterations. Nevertheless, the proposed scheme with tunable output corruption shows a higher resilience against approximate attacks than conventional point-function based techniques.

Following techniques [26], [98], [99] improves the Anti-SAT structure, i.e., two key-controlled sub-blocks, to achieve high SAT resilience as well as sufficient corruption rate.

Whereas the Anti-SAT block uses complementary sub-blocks, Noise-based logic locking [98] introduces non-complementary sub-blocks, whose outputs are XORed together. Such structure also avoids the output of the block having a probability skewed toward 0 or 1. By using non-complementary functions, a wide variety of structures, which are also less predictable to attackers, can be implemented.

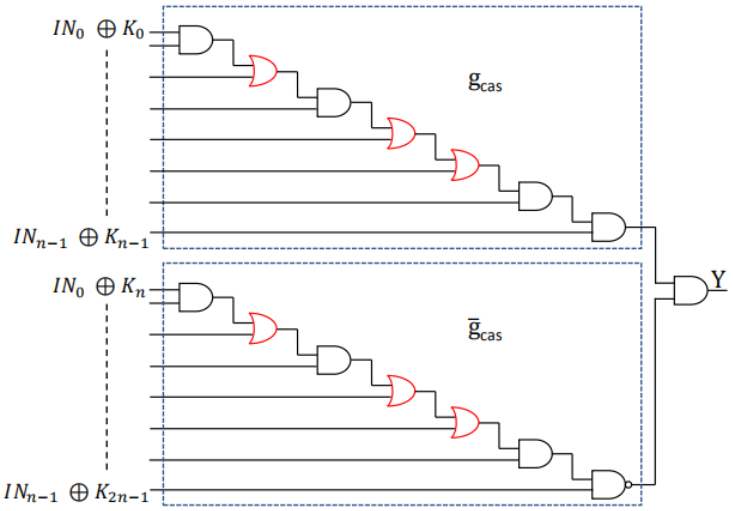


FIGURE 2.6: A possible structure of CAS-Lock [26].

CAS-Lock [26] proposes cascaded structures for the complementary sub-blocks. An example structure of CAS-Lock block is shown in Fig. 2.6. Instead of AND tree structures in Anti-SAT, the cascaded structures in this technique contain AND and OR gates. The proposed CAS-Lock block increases exponentially the complexity for SAT attack while providing considerable output corruption that helps thwart bypass attack [35]. Its output corruption is tunable by changing the location and number of AND/OR gates in the sub-blocks.

Generalized Anti-SAT (G-Anti-SAT) [99] introduces a generalized approach to designing SAT resilience logic lock. The work identifies a set of constraints for the function of each sub-block that can enable achieving maximum SAT resilience as well as non-trivial corruption. It then uses K-maps to implement such functions. A large variety of structures for sub-blocks can be realized, either complementary or non-complementary, AND tree or non-AND tree. Thus, Anti-SAT and CAS-Lock can be considered as special cases of G-Anti-SAT.

2.4.2 Corrupt-and-Correct Locking

Corrupt-and-Correct (CAC) locking scheme is illustrated in Fig. 2.7. In this scheme, the circuit function is altered by insertion of a perturb unit. A block controlled by key-inputs, termed restore unit, is used to restore the original functionality if only the correct key is applied. The advantage of CAC scheme is that even if the restore unit is removed (assuming that the attacker can detect the block by tracing the key-inputs), the remaining circuit is still functionally corrupted by the perturb unit. Similar to point-function based techniques, CAC prevents the SAT attack by exponentially increasing the number of iterations.

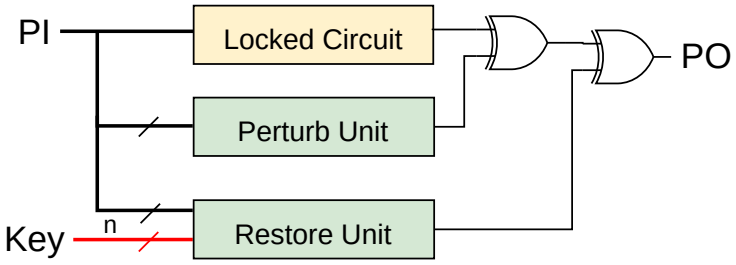


FIGURE 2.7: A general structure of Corrupt-and-Correct locking scheme.

Stripped Functionality Logic Locking (SFLL) [23]–[25], [100], [101] is a well known CAC method. The perturb unit is flipped for one or several input patterns, referred to as protected patterns, which are also the correct key values. Both perturb and restore units can be implemented as a point function [23]; in this case, there is only one protected pattern, hence the SAT resilience is maximum.

Several variants of SFLL have been introduced. SFLL-HD [24] allows tuning the output corruption. In SFLL-HD^h, a comparator structure block is added to strip the functionality of the original design. The block contains comparators and adders to calculate the Hamming distance between the current input and the hardcoded secret, and compare it to h (the Hamming distance between the input value and the correct key). Such block could incur significant overheads. By configuring more protected input patterns (i.e., increasing h in SFLL-HD^h), the output corruption of SFLL can be increased, however, at the cost of decreasing SAT-attack resilience. The FALL attack [102] was proposed to break SFLL-HD by identifying and analyzing the inserted perturb unit. It is a HD checker block with distinct properties and contains the hardcoded secret; moreover, its structure is left as is even after logic synthesis. Based on the unateness and HD properties of the HD checker, the attack is able to locate the perturb unit and extract the correct key.

SFLL-flex [24] allows user-defined protected patterns, which are hardcoded into the perturb unit. The restore units store these patterns into input cubes, which forms the secret key. SFLL-rem [100] introduces a method that strips functionality of the design by removing logic instead of inserting the perturb unit with hardcoded patterns. The technique identifies a net in the circuit that has at least n inputs in its fanin, where n is the key size. A stuck-at-fault is inserted in this net to remove the corresponding logic cone. The test patterns for this fault are generated, among which one is selected as the secret key value and is configured in the restore unit. Therefore, this method decreases overhead and prevents the FALL attack, compared to previous SFLL techniques.

CORALL [103] uses look-up tables to build the restore unit in order to increase output corruption without compromising SAT resilience. However,

due to the usage of LUTs, the key size may increase exponentially to the expected SAT-resilience level. Bilateral logic encryption [104] judiciously selects a logic cone in the circuit to lock and structurally obfuscate. M-CAS [26], a version of CAS-Lock, deploys two CAS-Lock blocks, where one with hardcoded secret key is the perturb unit and the other one controlled by key-inputs is the restore unit.

2.4.3 Crypto-based Locking

Cryptographic block ciphers are based on random permutations controlled by a secret key. Their security are formally proven against classical cryptanalysis methods as well as various attacks. Thus, block ciphers are indeed excellent candidates for SAT-attack protection on logic locking.

Crypto-based locking techniques consist in inserting cipher blocks (e.g., AES, PRESENT) controlled by key-inputs. Different from point-function based techniques which thwart SAT attack by increasing the number of iterations, these techniques hinder the attack by increasing the runtime for each iteration.

The work in [21] proposed to use an AES cipher block in addition to key-gates. The AES block has a fixed key value and its inputs are controlled by the key-inputs, whereas its outputs are used to control key-gates inserted in the circuit. It is shown that increasing the number of key-inputs connected to the AES block leads to exponentially increase in runtime of the SAT attack. However, the AES block incurs high overhead and is removable due to its distinct structure.

LoPher [22] introduces the idea of replacing gates with a block cipher. The technique uses PRESENT block cipher, which is based on S-Boxes, permutation layer and XOR layer. The S-box is generally a non-linear function, which contains XOR and AND gates. In fact, by fixing its inputs to specific values, different logic functions can be realized. Thus, gates in the circuit can be substituted with S-boxes controlled by key-inputs. Moreover, removing inserted S-boxes cannot recover the original netlist. Nevertheless, the technique has impractically high overheads.

2.5 Post-SAT Logic Locking Attacks

2.5.1 Oracle-Guided Attacks

Thanks to its efficiency, the SAT attack has been the foundation for various following attacks. These attacks may include additional steps or constraints, or use a more polyvalent solver.

Approximate & Bypass Attacks

Approximate attacks aim to find an *approximately* correct key, i.e., a key value for which output corruption of the locked circuit is very low. Notable approximate attacks are AppSAT [34] and Double DIP [95]. These attacks are effective on compound locking techniques [17], [18] where the key can be split into two parts, key bits dedicated to a SAT resilient technique (connected to the point-function lock) and the other bits dedicated to a traditional locking technique (connected to the key-gates). Hence, the goal is to discover the key bits belonging to key-gates because the remaining point-function lock has minimal output corruption. Compared to the SAT attack, approximate attacks terminate earlier without being trapped into solving the point-function lock.

The AppSAT attack [34], [97] adds to the SAT attack process the capability of corruptibility estimation and random query reinforcement. After each iteration, the attack deduce a key candidate. The attack regularly estimates the output corruptibility of the circuit with such key with random queries. The input-output pairs obtained from the oracle during random query are also added to the SAT solver as constraints for the next iteration. If the estimated corruptibility is repeatedly below a threshold, which is the corruptibility of the point-function lock, the attack terminates with an approximate key.

The Double DIP attack [95], during each iteration, finds a so-called 2DIP, a DIP that differentiates at least two wrong keys (instead of one in the original SAT attack). In this case, the miter circuit contains four copies of the locked netlist. 2DIPs exist because among the two eliminated wrong keys, one should has the wrong key bits for the key-gates and the other should have the wrong key bits for the point-function lock. Thus, when no more 2DIP can be found, all key bits of the key-gates are resolved and the attack terminates with an approximate key.

The Bypass attack [35] targets point-function based techniques by recovering the original circuit instead of finding the correct key. For that, it builds a bypass circuit to correct the output of the locked circuit. It can be observed that with a point-function lock, for each key value, different set of input patterns leads to corrupted outputs. Using the miter circuit and a SAT solver, the attack collects all disagreeing input patterns for two random wrong key values. After one of the key value is chosen, a bypass circuit is built to correct circuit outputs for these input patterns. On compound locking schemes, the attack can be followed an approximate attack to fully recover the original circuit.

Attacks with Advanced Solvers

Besides SAT solver, Satisfiability modulo theory (SMT) solver is a powerful solver which can process non-Boolean variables. Based on the same attack mechanism as the SAT attack, an attack based on SMT solver, termed SMT

attack, was proposed [36]. It can be considered as a superset of the SAT attack as it uses additional theory solvers to handle a more general class of constraint satisfaction problems and model more complex behavior. It can attack locking techniques based on physical behavior, which are non-applicable for the SAT attack, such as delay locking [105], which uses key-gates that change timing properties of the circuit. Furthermore, it can realize variants of the SAT attack such as approximate attacks.

Modern ATPG tools are technologically mature and contain advanced heuristics. An attack framework called CLIC-A [106] is based on ATPG tool. It proposes different methods targeting key-gate based techniques and SAT-resilient techniques. The key-input sensitization method is based on the Key sensitization attack [57]. The constraint-based ATPG method is another method that targets key-gate based techniques. An ATPG constraint function on signal lines asks the ATPG to generate test patterns that result in these signal values satisfying the function. The method solves logic cones individually. For each run, the ATPG generates a test for a fault at each key-input. Similar to the SAT attack, the oracle is queried with the test pattern and based on observed outputs, constraints are added for the next runs of ATPG. With additional constraints, the constraint function can be reduced using a logic minimizer and analyzed to deduce the key value. A method based on targeting key-dependent faults is used to counter SAT-resilient techniques. In these techniques, the protection logic (e.g., a point-function lock) has very few outputs, where typically all key-inputs converge upon, and contains the hardcoded secret. By generating the test patterns that can activate the faults at the output of this block, CLIC-A is able to find the hardcoded key by analysing the test patterns without access to an oracle.

Sequential SAT Attacks

The SAT attack works on combinational model and assumes the scan access on the oracle. Nevertheless, the scan chains may not be accessible. Sequential SAT-based attacks have been proposed to work on sequential circuit model [37], [107], [108]. They rely on unrolling the circuit and analyzing its Boolean properties with a model checker, which is equivalent to bounded model checking problem. The circuit is unrolled to a finite cycles; the unrolled circuit is hence combinational and can be used to construct a miter circuit. Similar to DIP in combinational attack, in each iteration, the solver finds a distinguishing input sequence. To avoid state space explosion problem when unrolling, the work in [107] introduced several termination conditions for the attack. The attack runtime was improved in [37] by dynamic simplification of key conditions. These attacks are shown to be effective on small low-depth circuits.

2.5.2 Oracle-Less Attacks

Removal Attacks

Removal attacks [38], [39], [96] aim to recover the original design by removing the inserted protection logic. These attacks can detect such structures based on their distinct structural features.

The structural vulnerabilities of point-function based techniques and their variants [17], [18], [23], [24] have been identified. Indeed, a point-function lock can be removed by detecting its output signal. The removal attack in [38] presents methods to identify such signal. One method is based on signal probability analysis. The output of a point-function lock has a highly skewed probability due to the AND tree. For n-bit point-function lock, the probability of this signal is $1/2^n$ [96]. Such skewed probability is rarely happened in a netlist and, hence, is singled out after signal probability analysis process. Even in the case of Anti-SAT with structural obfuscation (cf. Fig. 2.5(b)), a method based on fanin analysis can identify its output since all key-inputs converge at this signal. Furthermore, the point-function lock is structural isolated from the locked circuit and its size can be estimated, it can be detected using partitioning algorithms on netlist [17].

GNNUnlock [39] use graph neural network to facilitate identification of protection logic. The requirements for the attack are the locking algorithm and its setting/parameters, knowledge of key-inputs, the technology library and synthesis settings. The attack first identifies the specific and common characteristics of nodes in the protection logic blocks. The netlist is treated as a graph and each node is associated with a feature vector concerning its fan-in, fan-out and neighbors. After training, the graph neural network is able to classify which nodes belong to the protection logic blocks. The attack is able to detect the perturb unit used in CAC techniques (cf. Fig. 2.7), which was previously considered hard to detect.

Synthesis-based Attacks

Synthesis-based attacks [41], [109] aim to extract the secret key by synthesizing the locked netlist upon applying constraints on key-inputs. This method takes advantages of constant propagation in logic synthesis. Information extracted from the synthesis tool is used to correlate the structural properties of the circuit to the correct key.

Recent synthesis-based attacks are SWEEP [109] and SCOPE [41]. While SWEEP requires knowledge about the locking algorithm and training data, SCOPE is enhanced from SWEEP to be unsupervised since it does not require any training data and is oblivious of the locking algorithm. SCOPE targets each key-input individually. For a key-input, two copies of the locked netlist are created by assigning logic 0 and logic 1 to the key-input, while other

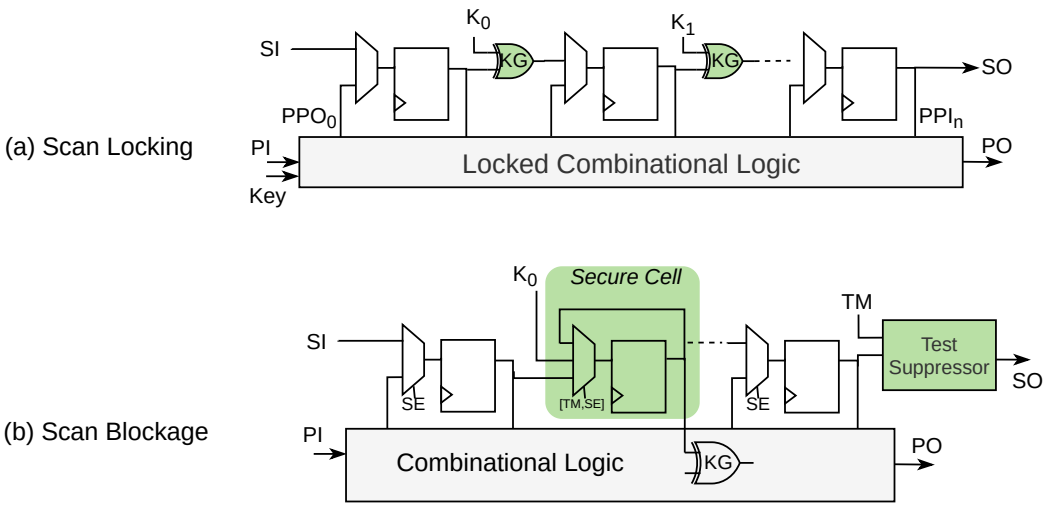


FIGURE 2.8: Example of scan chain-based defenses.

key-inputs are untouched. Both copies are then optimized with a logic synthesis tool. All available information concerning performance, topology and Boolean properties are recorded for comparison. The process is repeated for all key-inputs. A matrix is then formed with the comparison data in all features from all key-inputs. It is analyzed with a clustering process to perform the unsupervised labeling process on the key-inputs. Each key-input is labeled 0, 1 or undetermined. With this attack, logic locking techniques for which there is an observable structural difference when applying 0 or 1 to a key-input are vulnerable.

2.6 Defenses based on Scan Chains

Strong oracle-guided attacks on logic locking rely on the opportunity to control the inputs and to observe the outputs of the combinational block under attack via scan chains in the oracle. A generation of defenses based on scan chains for logic locking has been proposed and evaluated with attacks [110].

2.6.1 Scan Locking

One class of solutions based on the same principle as logic locking is referred to as scan locking. As illustrated in Fig. 2.8a, key-gates are inserted in the datapath of scan chains to corrupt scan data, both at shift in and shift out operations. The inserted logic is controllable so that authorized tester can access to uncorrupted test data. In this case, a scan access key is required to unlock the scan chains. The key-input of each key-gate can be static [27] or dynamic [28], [111], [112]. In the latter case, the key-inputs are controlled by a linear feedback shift register (LFSR); here the secret key is the seed of the

LFSR. Thus, key-gates are controlled dynamically and the scan response is different even for the same scan stimulus.

As is the case with logic locking, the output of locked scan chains reflect the inversion effect of key-gates Furthermore, inserted logic at the scan chains can be included in the combinational model of the circuit, as a combination of key-gates inserted at PPIs and PPOs. SAT-based attacks have been proposed to counter scan locking [113]–[115]. Static scan locking is also vulnerable to Sequential SAT attacks [37], [107], [108].

For dynamic scan locking, since the attackers can obtain the netlist of the circuit, they can have the polynomial function of the LFSR and deduce the equation corresponding to each cycle. SAT-based attacks in [114], [115] incorporate these equations into the locked circuit's combinational model and are able to recover the LFSR seed.

2.6.2 Scan Blockage

Scan blockage techniques take advantage of key-registers, i.e., dedicated registers for feeding logic locking key to the circuit. In this case, the key-registers are incorporated into scan chains. Upon using the scan chains in the oracle, the content of the key-registers is altered; hence, the circuit is not supplied with the correct key and becomes nonfunctional. Different from scan locking, these techniques do not involve a scan access key.

In [29], key-registers, referred to as Secure Cells (cf. Fig. 2.8b), can either hold its previous state, shift the content or input the key. In addition, a test suppressor is used to delete scan output data whenever the circuit switches from functional mode to test mode, hence, preventing leaking the key through scan data. Nevertheless, full test and debug facilities after production and delivering are provided. Manufacturing test can be performed without the correct key; key-registers are programmable via scan chains and can be assigned with values generated by ATPG tool. For debug and post-silicon validation where functional operation is required, a trusted tester can program the correct key into key-register via scan chains. However, due to testing support, the solution does not control the scan shift-in operation, which is exploited by a customized attack called Shift-and-Leak [116]. The key bits stored in key-registers can be shifted to other observable scan FFs and test patterns can be used to set a condition for sensitizing each key bit to POs.

Following solutions introduce a dedicated scan chain for key-registers to avoid shifting key bits to scan FFs. OraP [117] proposes to reset all key-registers when the oracle switches from functional mode to test mode. In other words, the circuit becomes locked when testing is applied. However, it is possible to scan out a correct response from the oracle circuit; it is the last response of the unlocked circuit operation, before the scan chains are activated. Such case allows oracle-guided attacks to partially recover the logic

locking key [30]. DisORC [30], an improved version of OraP, includes scan blocking circuitry to ensure that no correct response can be scanned out when key-registers are reset. It also allows control of the key-registers via JTAG to enable testing by untrusted testers. Another solution in [31] designs the key-register to include a trap storage whose the value cannot be shifted out, hence preventing Shift-and-Leak attack.

2.6.3 Scan Controlling

While scan locking targets solely on the scan data path, solutions based on scan controlling take into account the control signals, i.e., *Scan Enable, Clock, Test Mode*, of the scan chains. Likewise, a scan access key is required to unlock the scan chains.

The work in [32] introduces a key-based authentication for selected scan FFs. It modifies a scan FF with additional comparator-based circuitry that controls its Scan Enable signal. If the inserted scan access key is identical to the logic-locking key, the access to scan chains is granted. Otherwise, Scan Enable signals to modified scan FFs are inverted based on the initial scan content and the input test vector. Thus, the scan shift-in and shift-out operation are disrupted. The same authors also proposed a clock controller for the scan chains in order to freeze unauthorized scan operations [33]. For that, the clock signal for scan FFs is suppressed if a wrong scan access key is applied.

Chapter 3

Key-Gate Insertion Strategy for Effective Logic Locking

Contents

3.1 Introduction

With logic locking, high output corruption is desirable to guarantee function disruption upon the application of a wrong key value. Therefore, metrics for output corruption have been introduced [20], [85], [118]: output corruption rate presents the probability of observing erroneous bit(s) at the output vector of a locked circuit; output corruption coverage presents the magnitude of corruption propagated to circuit outputs; output corruptibility presents the probability of corruption at any circuit output.

Insertion strategy of protection logic significantly impacts output corruption, especially with key-gate based logic locking. Indeed, output corruption has been a major criterion for key-gate insertion strategies proposed in the literature.

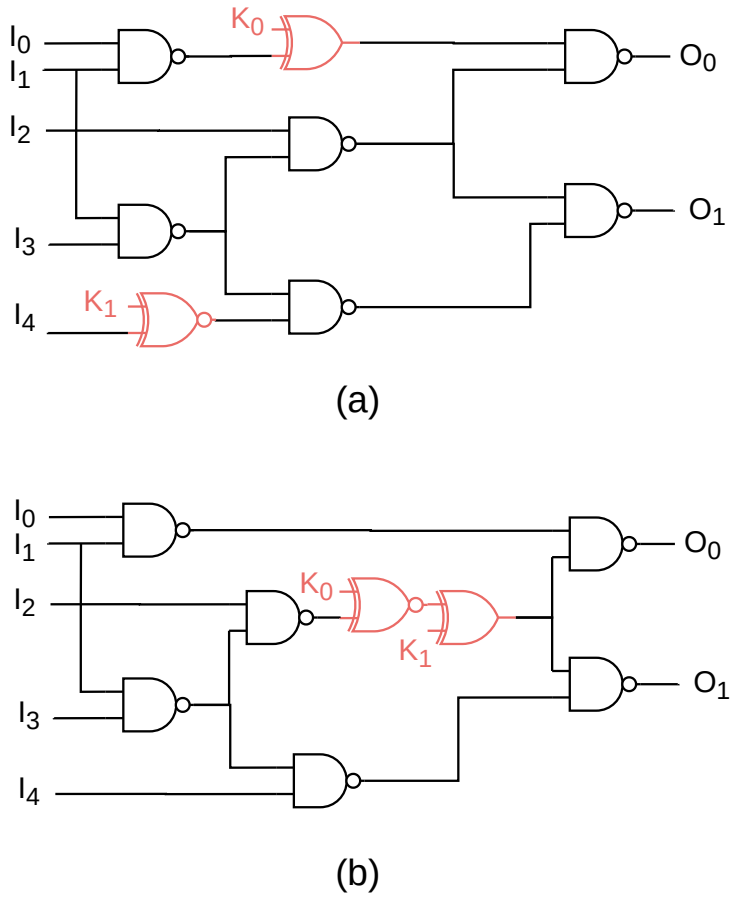


FIGURE 3.1: Key-gate insertion strategies. (a) RLL, (b) FLL.

The primitive technique Random Logic Locking (RLL) inserts key-gates randomly in the circuit netlist. More advanced techniques such as Fault-based Logic Locking (FLL) [19], [20] focus on improving output corruption. For example in Fig. 3.1, with RLL, each key-gate only has impact on one output; whereas with FLL, each key-gate impacts all outputs. Indeed, FLL [19], [20]

is the most effective strategy in terms of output corruption, reaching the optimal 50% output corruptibility. It considers a key-gate corrupting a signal equivalent to a fault appearing at such node. A so-called fault impact for each node is the product of the number of patterns that sensitize such fault and the number of affected outputs. The fault impact is measured by simulation with random patterns. To insert each key-gate, the strategy iteratively chooses the node with the highest fault impact. The execution time of the FLL strategy is estimated as:

$$T_{FLL} = t_{sim} \times N \times K \tag{3.1}$$

where t_{sim} is the simulation time to compute fault impact for each node, N is the total number of nodes and K is the number of key-gates. Note that N increases after each insertion of a key-gate. As shown in Equation 3.1, the strategy requires intensive simulation, which leads to scalability issue. In addition, as reported in [20], a large number of key-gates are inserted at the circuit outputs. As fault impact for each node is recomputed every time a key-gate is added, the chosen node with highest fault impact could be at the previously inserted key-gate. Thus, this leads to series of key-gates, which increase the number of correct key values. In the example in Fig. 3.1b, there are two correct values, $K_0K_1 = 10$ and $K_0K_1 = 01$, which increases the probability for an attacker to guess the correct key.

Succeeding techniques also deploy the fault impact metric, resulting in long execution time [87], [88]. The work in [89] proposed a more scalable strategy, which also presents a trade-off between execution time of insertion strategy and output corruption. In this work, the netlist is represented as a graph where each gate is a vertex and each wire is an edge in the graph. Centrality indicator is used to find the most significant vertices in the graph by measuring metrics such as closeness-centrality and between-centrality. Such nodes will be the locations for key-gates. However, this strategy only processes the netlist as a graph without taking into account the logical function of each gate, which explains why it takes less time to find key-gate locations than FLL. Nevertheless, its output corruption is substantially less optimal compared to that of FLL.

In this chapter, we propose a scalable Key-gate Insertion strategy based on Probability analysis (KIP) that is optimized for output corruption metrics. Nodes for insertion are chosen according to their output corruption score, which is computed by measuring the change in the probability of outputs. This metric also allows to avoid inserting key-gates in series. We demonstrate the strategy for XOR/XNOR key-gate insertion. We provide a comparison in terms of output corruption and overhead of the KIP strategy with FLL and RLL.

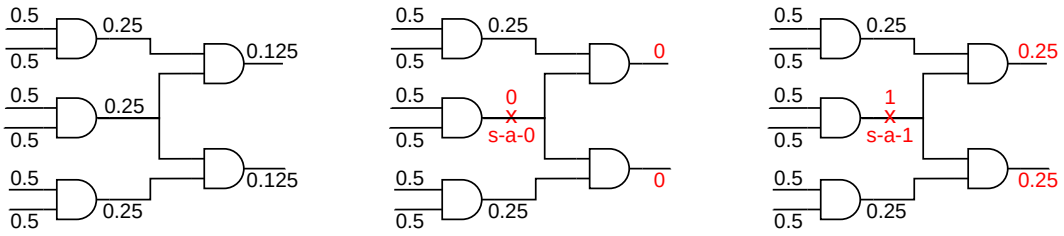


FIGURE 3.2: Change in probability of signals when stuck-at-faults appear.

3.2 Proposed Key-Gate Insertion Strategy

The KIP strategy consists in ranking nodes in the circuit based on score and selecting nodes for key-gate insertion.

3.2.1 Node Ranking

The principle of the proposed strategy is to rank every node of the circuit according to a metric called *output corruption score*. For each node, this metric reflects the impact on outputs if the signal in this node is corrupted due to the inserted key-gate. Similar to FLL, this strategy emulates the corruption by inserting stuck-at-faults at such node[1]. The impact of a given fault on outputs is measured by recording the difference in the outputs' probability (to be logic 1)[2] with and without the fault.

Calculating the probability of nodes in a netlist consists in propagating the probability of each node from the circuit inputs to the circuit outputs. First circuit inputs are assigned a probability of 0.5; for sequential circuits, the outputs of FFs are considered as PPIs, which are assigned the same probability as inputs. If the input signals of a gate are independent, the probability of its output depends on the probability of its input signals and its logical function. Nevertheless, netlists often contain convergent paths, where correlations among input signals of a gate appear. For such gates in a netlist, a different method is required to correctly measure the probability of its output [119]–[121].

A node with a s-a-f 0 or s-a-f 1 changes its probability to 0 or 1 respectively. Hence, the probability of nodes in its fan-out is influenced, as depicted in Fig. 3.2.

[1] Applying a wrong key to a XOR/XNOR key-gate is equivalent to the activation of a s-a-f. Either a s-a-0 or s-a-1 will get triggered.

[2] For the rest of the chapter, we use probability of a node to refer to its probability to be logic 1.

Score Calculation

The calculation of output corruption score for each node consists in computing the probability of outputs. Firstly, the probability of outputs in the original circuit is measured. Then, s-a-1/0 is inserted at that node and the probability is recomputed. By comparing the probabilities before and after the fault insertion, one obtains the total probability difference Δp_{saf} and the number of outputs that have their probability changed n_{saf}. Δp_{saf} is the sum of absolute probability difference of each circuit output:

$$\Delta p_{saf} = \sum_{i=1}^{n_O} |p_i - p_{i_{saf}}| \tag{3.2}$$

where n_O is the number of circuit outputs.

The output corruption score is calculated as:

$$S_{OC} = \Delta p_{sa0} \times n_{sa0} + \Delta p_{sa1} \times n_{sa1} \tag{3.3}$$

Inserting key-gates at nodes with high score will impact most of the outputs for most of the input patterns, resulting in high output corruption coverage and corruption rate. One can notice that nodes at circuit outputs always have S_{OC} of 1 since there is only one output affected and the total probability difference due to s-a-0 and s-a-1 $\Delta p_{sa0} + \Delta p_{sa1}$ is 1. On the other hand, internal nodes with large fan-out potentially have high S_{OC} score and are favored by the strategy to achieve higher output corruption coverage.

3.2.2 Algorithm

Algorithm 1 describes the KIP strategy, using for XOR/XNOR key-gate insertion. The algorithm includes two steps, ranking nodes and selecting nodes for insertion. The output corruption score of each node is calculated and nodes are ranked according to their score in a descending order. After, node selection starts from the node with highest score. Here an additional criterion is applied; the nodes that have the same score as the previously chosen node will not be selected for key-gate insertion. This is because nodes that have the same score are close to each other, such as nodes connected by a buffer. Thus, selecting only one among these nodes avoids series of key-gates.

Scalability

The execution time of the KIP strategy is essentially the node ranking step. It can be estimated as:

$$T = t_{prob} \times N \tag{3.4}$$

Algorithm 1: The KIP strategy

Data: netlist, keySize
Result: Locked netlist

```
1  nodeList = [inputs, signals, outputs]
2  NbInsertedKeyGates = 0
3  for I in nodeList do
4  |   Calculate S_OC score of I
5  end
6  rankedNodeList = rank(nodeList, descending order based on S_OC)
7  while NbInsertedKeyGates < keySize do
8  |   Node, Score = rankedNodeList.pop(0)
9  |   if Score ≠ previousScore then
10 |   |   Insert a XOR/XNOR key-gate at Node
11 |   |   NbInsertedKeyGates += 1
12 |   else
13 |   |   continue                    ▷ To avoid series of key-gates
14 |   end
15 end
16 return locked netlist
```

where t_{prob} is amount of time for calculating output corruption score of a node, N is the number of nodes.

In comparison with FLL, FLL redoes the ranking each time a key-gate is inserted (cf. Equation 3.1), whereas KIP only ranks nodes once. Therefore, our strategy is more scalable than FLL.

3.3 Experimental Results

For measuring probability, we used Signal Probability Reliability Analysis (SPRA) tool [122]. In our experiments, we chose not to use the option that takes into account reconvergent paths. Without this option, the program takes much less time, despite slightly less accurate measurement, than with this option [119], [120]. Nevertheless, as shown in Section 3.3.2, optimal results for output corruption were obtained.

3.3.1 Runtime Evaluation

To assess the runtime of KIP, we implemented it on ISCAS'85, MCNC, ISCAS'89 and ITC'99 benchmarks. The experiments were executed on an 8-core Intel processor running at 1.90GHz with 16 GB RAM.

Table 3.1 shows the execution time results in an increasing order. In general, larger circuits, i.e., circuits with higher number of nodes (cf. second column),

TABLE 3.1: RUNTIME OF KIP STRATEGY ON BENCHMARKS

Bench	Nb nodes	Runtime (s)
s510	236	7.58
s641	433	16.29
s838	457	19.57
c5315	2485	225.35
i8	2597	205.15
s5378	3050	495.23
seq	3560	469.87
c7552	3720	605.5
apex4	5370	1609.4
des	6729	3340.83
s9234	5844	3729.66
s13207	8729	12351.7
b15_C	8922	14337.47
b14_C	10098	19663.67
s15850	10397	25355.29

require more time than smaller ones; however, it is a non-linear relation; the runtime scales up faster than the size of the circuit. This is because the probability measurement runtime depends on the circuit size. Nevertheless, for small circuits, the KIP strategy finished in a matter of minutes. For example, with benchmark c7552, it finished in 10 minutes; whereas the FLL algorithm "took two hours to encrypt the c7552 circuit" [20][3].

3.3.2 Output Corruption Evaluation

We evaluated the output corruption of KIP with metrics including output corruptibility, output corruption rate and output corruption coverage. We implemented XOR/XNOR key-gate insertion with our strategy, FLL and RLL, each on six benchmark circuits. For each benchmark, the number of inserted key-gates is 5% of the number of gates in the circuit (130 for i8, 124 for c5315, 178 for seq, 186 for c7552, 269 for apex4, 336 for des).

For this evaluation, each circuit was simulated with 100 wrong key values, each with 1000 random input patterns. The results are presented in Fig. 3.3. As KIP and FLL are optimized for output corruption, both achieve optimal results in all metrics. For the KIP strategy, most circuits achieved output corruptibility from 40% to the optimum 50%. Its output corruptibility is equivalent with that of FLL; the results are slightly better for four circuits. It also has maximum output corruption coverage due to the fact that it favors nodes that effect the most outputs as possible; the results are better than that of FLL for two circuits. It achieves 100% corruption rate in all circuits, which is equal

[3]The paper [20] only reported the runtime of FLL for benchmark c7552.

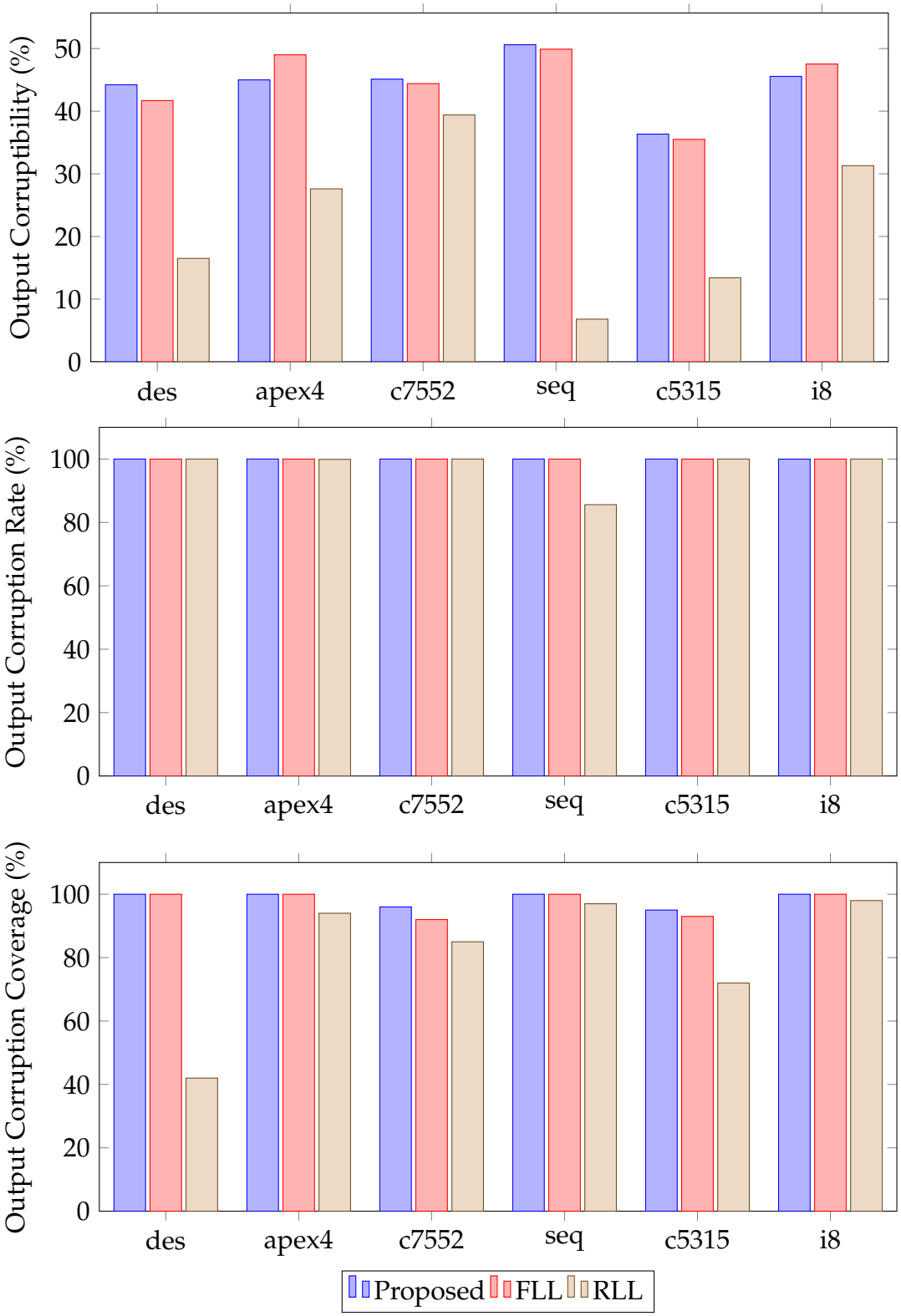


FIGURE 3.3: Output corruption evaluation and comparison with FLL, RLL. Benchmarks are in decreasing size order.

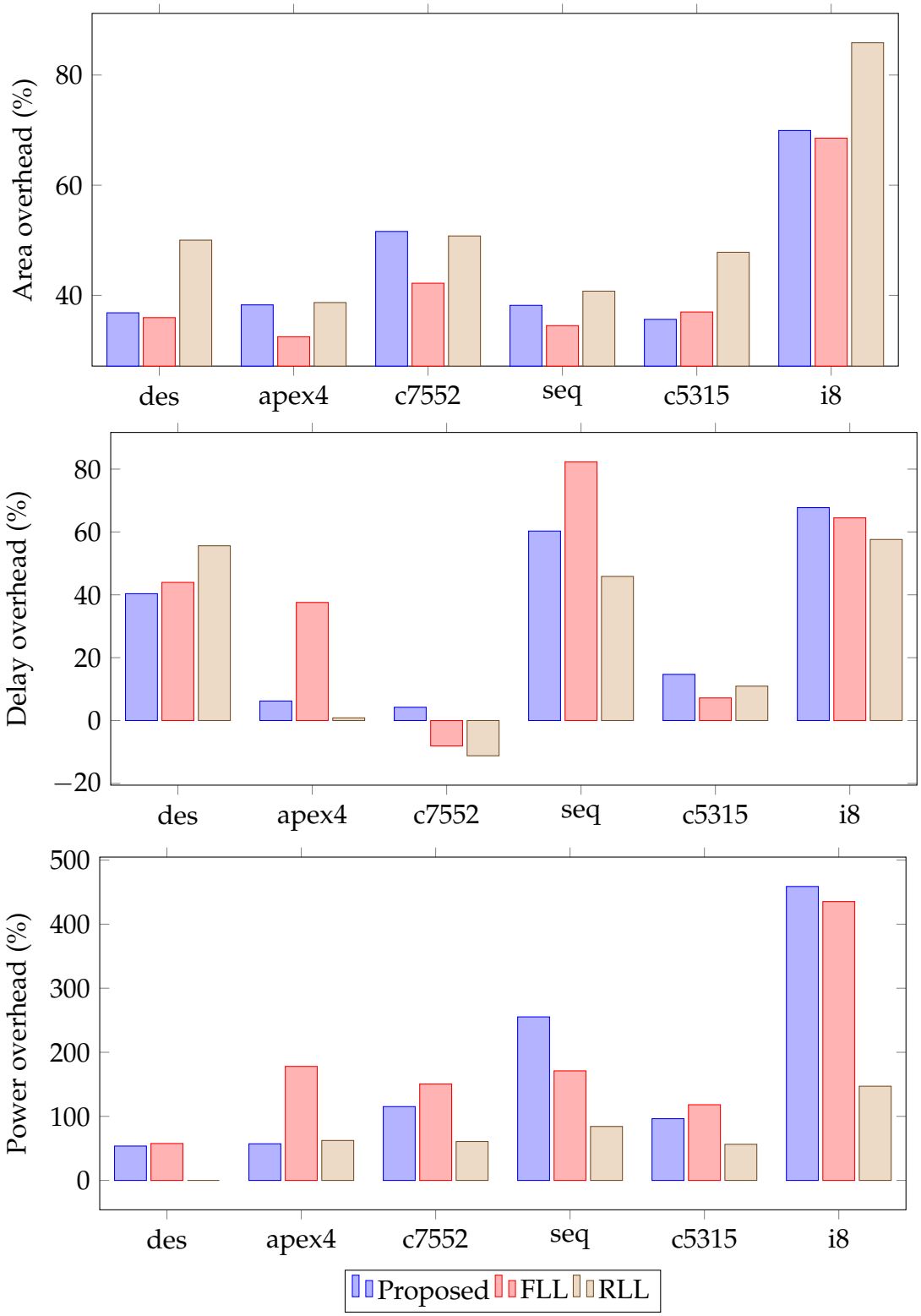


FIGURE 3.4: Overhead evaluation and comparison with FLL, RLL. Benchmarks are in decreasing size order.

to FLL and better than RLL for one circuit. KIP performs significantly better than RLL, especially in output corruptibility and corruption coverage. The results imply that KIP is as efficient as FLL in terms of output corruption, with significantly shorter execution time.

3.3.3 Overhead Evaluation

We measured the overhead of circuits used in the previous evaluation. The circuits were synthesized with Synopsys Design Compiler using a 65nm technology. Fig. 3.4 shows the results in term of area, delay and power overhead. The average overhead of KIP is equal to that of FLL, which is smaller in average area and larger in average delay compared to RLL. To limit the delay overhead of KIP, an additional constraint for avoiding inserting key-gates in the critical path [53] can be incorporated in the strategy. The average power overhead of KIP, as well as that of FLL, is high in these simulation results since the inserted key-gates lead to high output corruption, hence higher switching in the circuit. However, once the IC is activated, key-inputs are fixed and the key-gates cannot cause additional switching.

3.4 Conclusion

In this chapter, we present KIP, a key-gate insertion strategy aimed for high-output-corruption logic locking. Its output corruption score metric based on measuring the probability of outputs allows finding key-gate locations where corruption is highly observable at several outputs. This metric also prevents inserting key-gates in series. Demonstrated with XOR key-gate insertion, it achieves optimal results in all output corruption metrics. Compared to FLL, KIP requires much less execution time. The runtime of KIP can be further optimized by only considering nodes in selective regions which has high influence on the circuit's function such as the controller.

Locking a circuit only with this insertion strategy lacks necessary security measures. Powerful attacks like the SAT attack can be hinder with more elaborated protection logic as shown in the following chapter.

Chapter 4

Building Secure and Effective Logic Locking with Switchable Key-Gates

Contents

4.1 Introduction

Key-gate based logic locking is highly effective in terms of output corruption when using a judicious insertion strategy such as the KIP strategy in Chapter 3, as well as FLL [20]. Nevertheless, due to simple key-gate structure (e.g, two-input XOR/XNOR), key-gate based techniques [20], [53], [56], [57], [123] have been shown to be highly susceptible to the SAT attack [16][1]. As shown in Table 4.1, evaluated against benchmarks locked with XOR key-gates using KIP strategy, the attack is able to return the correct key with a few iterations in just seconds.

TABLE 4.1: SAT ATTACK ON A XOR KEY-GATE LOGIC LOCKING TECHNIQUE USING KIP STRATEGY

Bench	Key size	Nb iterations	Runtime (s)
i8	130	32	2.62
c5315	124	10	0.72
seq	178	38	7.1
c7552	186	10	1
apex4	269	92	5.36
des	336	19	5.78

Primitive point-function based logic locking provides a provable protection against the SAT attack, which exponentially increases the number of iterations [17], [18], [24], [94]. However, regarding these techniques, a fundamental trade-off has been identified between SAT resilience and output corruption [124]. Indeed, to ensure high SAT resilience, the output corruption is greatly reduced such that the circuit is mostly functional despite being supplied with a wrong key value [125].

In this chapter, we propose a novel secure logic locking technique, SKG-Lock, that aims to thwart SAT-based attacks while maintaining significant output corruption. The proposed provable SAT-resilience scheme is based on the novel concept of decoy key-inputs and switchable key-gates (SKG). We present two architectures of SKG-Lock and provide a proof for SAT resilience for both architectures. We revise a framework of SKG-Lock where SKGs can be inserted using the KIP strategy proposed in Chapter 3. We analyze the security of the SKG-Lock framework against different attacks. We provide evaluations of attack resilience, output corruption and overhead of SKG-Lock, along with comparisons with state-of-the-art techniques.

The rest of the chapter is organized as follows. Section 4.2 introduces the fundamental components of the proposed logic locking scheme, SKG-Lock. Section 4.3 presents a light-weight SAT-resilient version of SKG-Lock. Section 4.4 details the SKG-Lock framework and its security analysis against various

[1]The success of the SAT attack is also due to the presumed open scan access in the oracle.

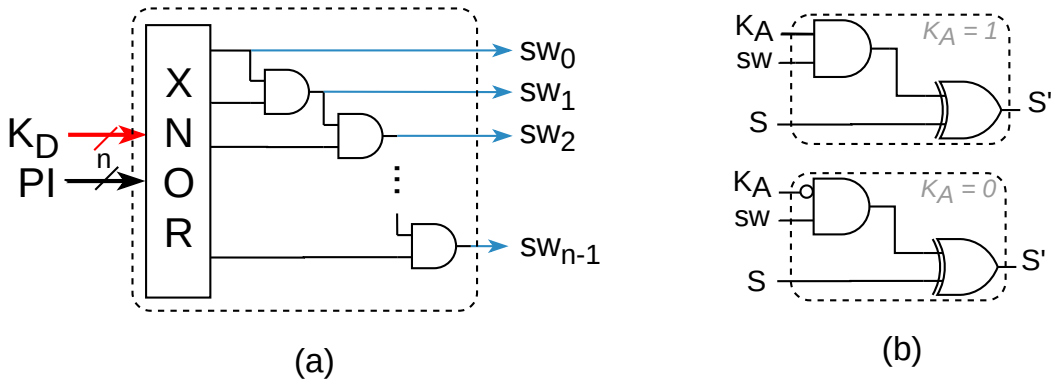


FIGURE 4.1: Basic structure of SKG-Lock components. (a) Switch controller. (b) Switchable key-gates.

attacks. Experimental results for the evaluations of security, output corruption and overhead are shown in Section 4.5. Finally, Section 4.7 concludes the chapter.

4.2 Components of SKG-Lock

The two fundamental components of SKG-Lock are *switchable key-gates* and a *switch controller*, as depicted in Fig. 4.1.

A switchable key-gate (SKG) has three inputs, two control signals — key-input K_A and switch-signal sw — and one signal S from the locked circuit (cf. Fig. 4.1.b). Hence, compared to a traditional XOR key-gate, in an SKG, the additional *switch-signal* enables the control of its corruptibility. To make an SKG corrupt the signal S, both of its control signals have to be asserted. Therefore, corruption only happens when an incorrect value is inserted at the key-input K_A and logic 1 is set on the switch-signal (in case of SKGs with positive switch[2]).

The switch controller (SWC) controls the switch-signals of the SKGs. A general design for an SWC is a comparator, constructed with a row of XNOR gates and a cascade of AND gates (cf. Fig. 4.1.a). Its inputs are n-bit key-input K_D and n-bit circuit inputs[3]. This SWC structure produces n switch-signals, each from each node in the AND cascade; for example, sw_{n-1} comes from the output of the last AND gate in the cascade, sw_{n-2} comes from the previous AND gate and so on.

Two sets of key-inputs, the *Activation Key* (K_A) and the *Decoy Key* (K_D) are introduced:

[2] An SKG with negative switch can be constructed with an OR gate and an XNOR gate.

[3] Circuit inputs can include primary inputs and pseudo primary inputs (outputs of flip-flops in a sequential circuit).

- K_A is connected to the SKGs.
- K_D is connected to the SWC.

Since SKGs are the components that cause corruption, the circuit is unlocked by inserting the correct K_A for each SKG, irrespective of the K_D value. Nevertheless, K_D is important for SAT resilience, as detailed in Section 4.4.3. Upon applying SKG-Lock to a design during the design phase, the designer sets a secret value for K_A. Note that both K_D and K_A come from a protected memory and are physically indistinguishable. Both are controllable key-inputs in the locked netlist used for oracle-guided attacks.

Definition 5 (Corruptibility of SKG) *Corruptibility is the probability of an SKG corrupting its insertion signal if a wrong key is supplied to its K_A key-input.*

The corruptibility of an SKG equals to the probability (to be logic 1) of its switch-signal. Multiple switch-signals can be outputted from the SWC, one from each node in the AND cascade. Hence each switch-signal has a different probability. The output at the end of the cascade sw_{n-1} is essentially the output of a point-function between K_D and PI. Therefore, sw_{n-1} presents low corruptibility ($1/2^n$) but maximal complexity for the SAT attack, as shown in the following section. Other switch-signals from the upstream of the cascade have higher probability. Thus, SKGs driven by them have higher corruptibility. The corruptibility C of each SKG controlled by a different switch-signal is

$$C_{sw_0} = \frac{1}{2}$$
$$C_{sw_1} = \frac{1}{2^2}$$
$$\dots$$
$$C_{sw_{n-1}} = \frac{1}{2^n}$$

4.3 Light-Weight SKG-Lock

4.3.1 Architecture

Fig. 4.2 depicts the light-weight SAT-resilient version of SKG-Lock, referred to as SKG-Lock^{lw}. It contains an SWC and an SKG. The SKG, inserted at a circuit output, is controlled by the switch-signal sw_{n-1}. We show subsequently that this structure achieves an n-secure SAT resilience level.

4.3.2 Security against SAT Attack

Notations & Assumptions: Without loss of generality, we can assume that the size of K_D and circuit PI is n; the correct value of each bit of K_A is 1 for every

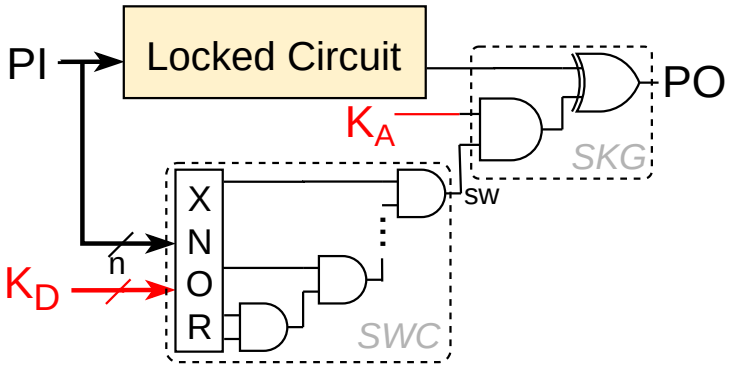


FIGURE 4.2: Light-weight SKG-Lock.

SKG; and each SKG is inserted at a different circuit output so that any corruption is observed at an output. A DIP produced at i-th iteration by the SAT attack is denoted as X_i. Let's denote N as the number of iterations of a SAT attack.

Proof for SAT resilience: Wrong key values that can be ruled out by a DIP X_i satisfying the following condition:

$$(K_A = 0) \wedge (\vec{K_D} = \vec{X_i}) \tag{4.1}$$

For any given X_i, there is one way to select K_A and one way to select K_D to satisfy the condition in Equation 4.1. Thus, each iteration identifies only one wrong key value. Hence, the number of iterations required by the SAT attack to eliminate all (2^n) wrong key values is $N = 2^n$. The circuit is n-secure against SAT attack.

4.3.3 Analysis

Compared to point-function based logic locking techniques such as SARLock and Anti-SAT, SKG-Lock^{lw} achieves the same SAT resilience level.

In terms of hardware overhead, SKG-Lock^{lw} presents an advantage since it consists of only one comparator. In comparison with point-function based techniques such as SARLock and Anti-SAT, for the same SAT resilience, the number of inserted gates for each technique is estimated as:

- SKG-Lock^{lw}: $n + 1$ XOR/XNOR gates, $n + 1$ AND gates
- SARLock: $n + 1$ XOR/XNOR gates, $2 \times n + 1$ AND/OR gates
- Anti-SAT: $2 \times n + 1$ XOR/XNOR gates, $2 \times n + 1$ AND/OR gates

SKG-Lock^{lw} requires half the number of gates compared to Anti-SAT.

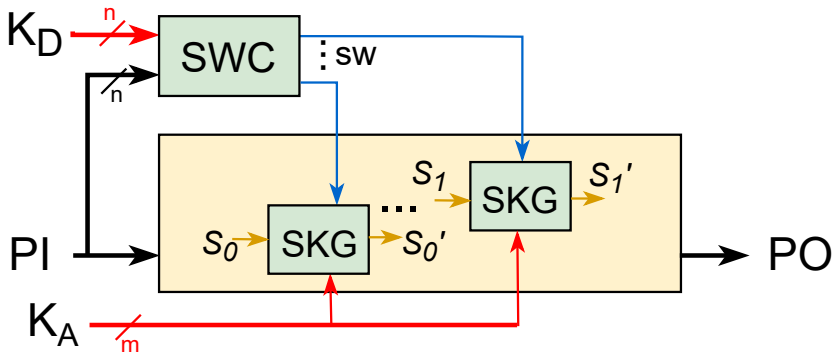


FIGURE 4.3: Architecture of SKG-Lock framework.

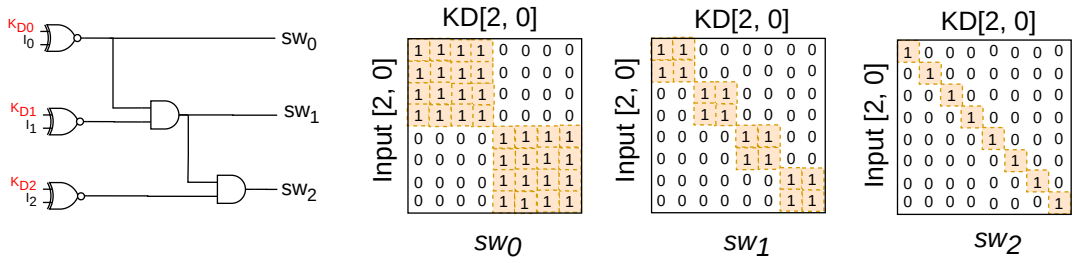
Similar to point-function based logic locking, SKG-Lock^{lw}, however, has remaining drawbacks: as K_A is only 1 bit, the ratio of correct keys over key space is 1/2; the low output corruption rate ($1/2^n$) and corruption coverage (it only causes corruption on one PO) issues remain; its structure is isolated from the locked circuit structure. By taking advantages of inserting multiple SKGs with different corruptibility, the SKG-Lock framework presented in the following section addresses all of these issues while achieving the maximum SAT resilience.

4.4 SKG-Lock Framework

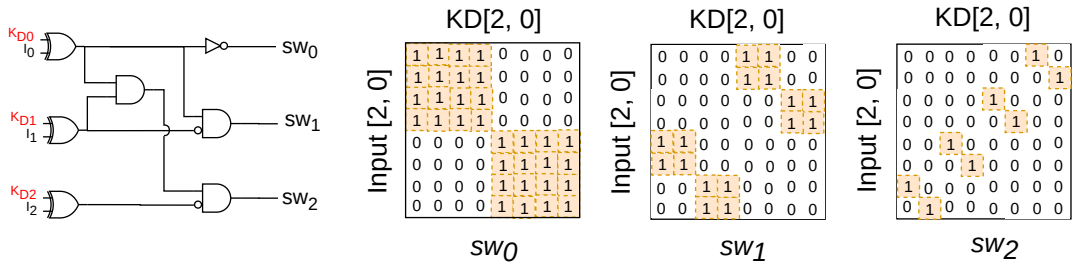
4.4.1 Architecture

The architecture of the SKG-Lock framework is illustrated in Fig. 4.3. Different from the light-weight version, it includes several SKGs controlled by different switch-signals. m SKGs (hence m-bit K_A) are inserted in the locked circuit. An SWC, with n-bit circuit inputs and n-bit K_D as its inputs, produces n switch-signals. In the case where $m = n$, each switch-signal controls one individual SKG. We will show that this SKG — switch-signal mapping is better for achieving both high SAT-resilience and output corruption than mapping one switch-signal to several SKGs in Section 4.5.1. In the case where $m \neq n$, several SKGs may be driven by the same switch-signal or certain switch-signals may be unconnected. The ratio of correct keys over key space of SKG-Lock is $1/2^m$.

Note that the use of several switch-signals and SKGs creates multiple connections between the SWC and the locked circuit. Thus, our SKG-Lock architecture, in its nature, achieves structural entanglement without requiring any compound structural obfuscation technique, such as wire entanglement used in Anti-SAT [94].



(a)



(b)

FIGURE 4.4: Examples of 3-bit SWC and the truth tables of switch-signals based on: (a) the structure in Fig. 4.1.a, (b) the structure used in SKG-Lock framework that avoids overlapping in the input patterns that activate each switch-signal.

SWC Structure

In the SKG-Lock framework, each switch-signal is used to control an SKG. However, with the SWC structure in Fig. 4.1.a, there is overlap in the input patterns that activate each switch-signal, and there are a large number of input patterns for which there is no corruption. Therefore, it is necessary to improve such SWC structure in order to increase the number of patterns for which there can be corruption. An example of the SWC structure used in the SKG-Lock framework is illustrated in Fig. 4.4(b). In the truth tables of switch-signals in Fig. 4.4, the rows present the primary inputs possibilities and the columns the key values possibilities. With the original structure in Fig. 4.4(a), due to overlap of input patterns that result in '1' for each switch-signal (cf. light orange boxes in the truth table), there can only be half of all patterns (4 out of 8) that can lead to corruption. Whereas with the improved structure in Fig. 4.4(b), different input patterns assert different switch-signals; hence, there are 7 out of 8 patterns for which there can be corruption. In case when all inserted K_A bits are incorrect, one can expect output corruption rate upto almost 100%, as shown in Section 4.5.2.

Compared to the original structure, this improved SWC structure requires more area; for n-bit SWC, it uses additional $n-2$ AND gates.

The estimated area overhead of the SKG-Lock framework is:

- n-bit SWC: n XOR/XNOR gates, $2 \times n - 2$ AND gates
- m SKGs: m XNOR/XNOR gates, m AND gates

4.4.2 Locking Algorithm

SKG Insertion

Since SKG-Lock provides security against SAT-based attacks (as shown in following section), the criteria for SKG insertion strategy is to maximize output corruption. As detailed in Chapter 3, our KIP strategy and the FLL strategy [20] provide optimum results with regard to output corruption metrics. These insertion strategies can be integrated in the SKG-Lock framework as follows. With the FLL strategy, since it is used for XOR key-gate insertion, it first inserts XOR key-gates in the circuit then substitutes XOR key-gates with SKGs. With the KIP strategy, since it ranks nodes based on output corruption score S_{OC}, SKGs are inserted at nodes with highest scores; this principle can also be used to define the mapping between switch-signal and SKG.

Algorithm 2: SKG-Lock locking algorithm

Data: Netlist, size of K_A (m), size of K_D (n), SKG insertion strategy
Result: Locked netlist

1 *Locked netlist* = *Netlist*;
2 Add n-bit K_D and m-bit K_A to *Locked Netlist* primary inputs;
3 Insert n-bit SWC block in *Locked Netlist*;
4 Map K_D and n chosen primary input to SWC inputs;
5 *insertionNodes* = insertionStrategy(*Netlist*, m);
6 **for** ***each*** *bit* K_{A_i} ***in*** K_A **do**
7 *node* = *insertionNodes*[i];
8 Insert an SKG at *node* in *Locked Netlist*;
9 Map K_{A_i} and sw_i to control inputs of the SKG;
10 **end**
11 Return *Locked netlist*;

Algorithm

Algorithm 2 describes how a netlist is locked with SKG-Lock. The designer defines:

- The key sizes:
  - The size of K_A, hence the number of SKGs,
  - The size of K_D, hence the size of SWC and the SAT-attack resilience level,

- The SKG insertion strategy and the rules to map switch-signals to SKGs and circuit inputs to the SWC.

The mapping rules for switch-signals to SKGs and circuit inputs to the SWC can be random or strategical depending on the SKG insertion strategy. With the KIP strategy, the rules are:

- the highest corruptibility switch-signal is mapped to the SKG inserted at node with highest S_{OC} score and so on;
- the inputs with lowest S_{OC} scores are connected to the SWC.

In term of execution time of the algorithm, the insertion process of the protection logic is fast; whereas the process for selecting SKG locations can be time-consuming, which depends on the chosen strategy.

4.4.3 Security against Oracle-Guided Attacks

Key Sensitization Attack

The key sensitization attack [57] is able to counter key-gate based logic locking by targeting each key-gate and sensitizing individual key-input. With SKG-Lock, for each SKG, there is a convergence path between its K_A signal and its switch-signal. Furthermore, switch-signals are convergent with (one to) several K_D signals. Therefore, there is an interference between each K_A key-input and K_D key-input(s), which prevents the attack from sensitizing individual key-input to a circuit output.

SAT Attack

Proof for SAT resilience: We consider the case where the circuit is locked with two SKGs: one SKG is driven by sw_{n-1} and another SKG is driven by sw_{n-2}. The same assumptions as mentioned in Section 4.3.2 are used.

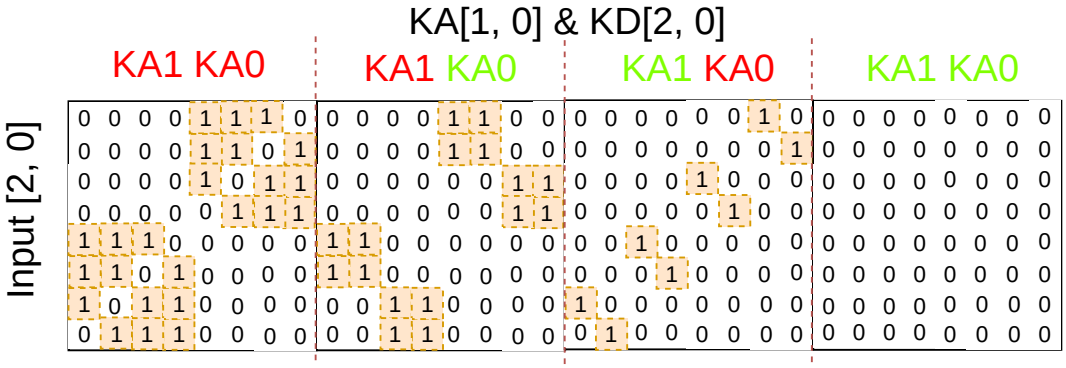
The condition for any wrong key value to be identified by a given X_i is:

$$\begin{aligned}\big[(\vec{K_A}[0]=0, \vec{K_A}[1]\in \mathbb{B}) \wedge (\vec{K_D}=\vec{X_i})\big] \vee \\ \big[(\vec{K_A}[0:1]=\vec{10}) \wedge (\vec{K_D}[0:n-2]=\vec{X_i}[0:n-2])\big]\end{aligned} \quad (4.2)$$

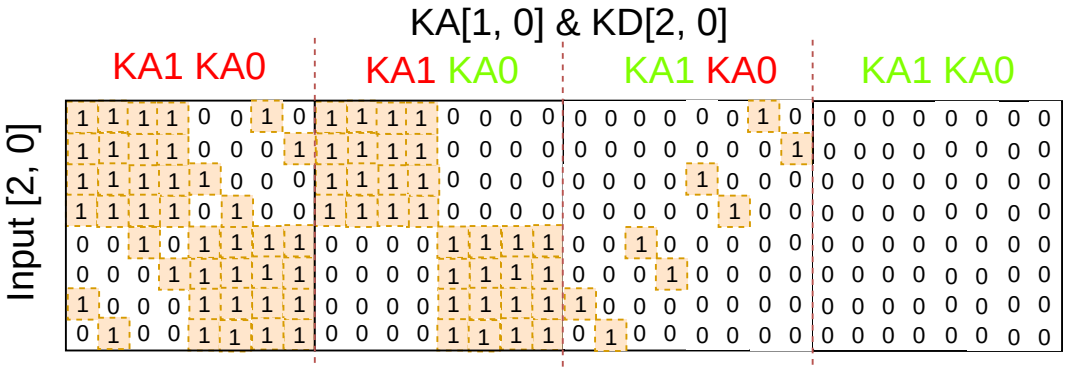
Thus, when $\vec{K_A}[0]=0$, the set of wrong key eliminated by X_i has the following form:

$$(\vec{K_A}[0]=0, \vec{K_A}[1]\in \mathbb{B}, \vec{K_D}=\vec{X_i}) \quad (4.3)$$

There is a one-to-one matching between K_D and X_i. Thus, any input pattern can be selected as a DIP to identify a unique set of wrong keys in the form of



(a) one SKG connected to sw_2 and one SKG connected to sw_1



(b) one SKG connected to sw_2 and one SKG connected to sw_0

FIGURE 4.5: The truth table representing corruption (1 in light orange) depending on the possible key values (correct bits in green and incorrect bits in red)

(4.3). Therefore, the total number of SAT iterations is $N = 2^n$. The circuit is n-secure against SAT attack.

Fig. 4.5 illustrates this proof with two examples of truth tables. In these examples, there is a 2-bit K_A, connected to two SKGs, and a 3-bit K_D, connected to a 3-bit SWC. Each truth table is divided into four sub-parts according to the four possible values of K_A (and in each sub-part, eight possible values for K_D). The two control signals of the first SKG are $K_A[0]$ and $sw_{n-1} = sw_2$. The two control signals of the second SKG are $K_A[1]$ and sw_1 in Fig. 4.5(a) or sw_0 in Fig. 4.5(b).

In both figures, the point-function behavior appears when $K_A[0]$ is not correct but $K_A[1]$ is correct (third sub-part of the truth tables), which imposes on the SAT attack at least $2^n (n = 3)$ iterations to eliminate all wrong key values in this sub-part. Besides, when $K_A[1]$ is not correct (first and second sub-part), the corruption[4] is increased, depending on which switch signal is used.

[4]These truth tables present the corruption at the signals on which the SKGs are inserted.

More generally, the presented proof holds for the case where there are more than two SKGs. n-secure SAT-resilience level is achieved as long as there is at least one SKG connected with sw_{n-1}. Similar to SKG-Lock^{lw}, the achieved SAT resilience depends on the size of the SWC.

AppSAT Attack

The AppSAT attack [34] aims to find an low-corruption key, i.e., a key value for which the output corruption of the locked circuit is very low. The work in [126] discovered that the AppSAT attack is not effective against logic locking techniques where different key values correspond to different amounts of output corruption. This is true for SKG-Lock since each key value indicates different set of incorrect key bits for SKGs and each SKG has a different corruptibility. The key value that results in the lowest corruption is the one in which only the key bit of the lowest corruptibility SKG is incorrect. Therefore, one can expect that AppSAT on SKG-Lock returns a key value that has several wrong key bits, which leads to considerable corruption.

Bypass Attack

The Bypass attack [35] aims to construct a bypass circuit to correct the corrupted outputs of a locked circuit. The attack builds a miter circuit with two copies of the locked circuit applied with two random key values. The miter circuit is used to find all input patterns that cause corrupted outputs. In SKG-Lock, two random key values may contain same wrong key bits for SKGs. Hence, the two locked copies may have the same wrong outputs for several input patterns, which then would go unnoticed by the attack. Furthermore, an incorrect key value could lead to significant output corruption rate and coverage, which results in an impractically large bypass circuit. Therefore, the attack is not efficient against SKG-Lock.

4.4.4 Countermeasures against Oracle-Less Attacks

The SKG-Lock framework, as it is, is susceptible against potential oracle-less attacks [38], [41]. In this section, we propose countermeasures for SKG-Lock against such attacks. These solutions consist of light-weight structural addition to the SWC and the SKGs.

Removal Attack

The probability-analysis based removal attack [38] uses signal probability analysis to detect the inserted protection logic. In SKG-Lock, the SWC, whose structure is similar to a point-function, generates a few switch-signals with highly skewed probabilities, especially sw_{n-1}. Thus, these switch-signals are detectable to the attack. To render the probabilities of these signals balanced,

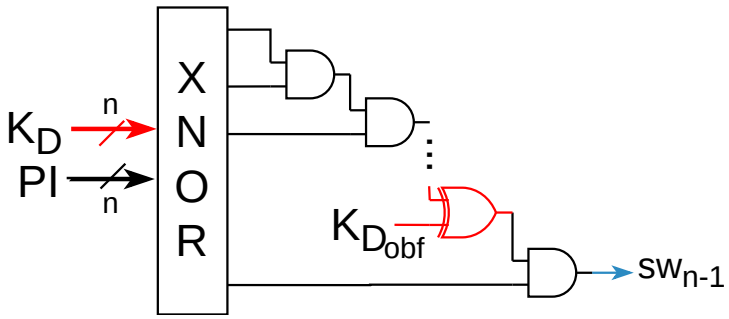


FIGURE 4.6: Removal attack countermeasure based on obfuscating the SWC with XOR key-gates.

XOR gates with additional key inputs $K_{D_{obf}}$ can be added to the AND cascade of the SWC, as illustrated in Fig. 4.6. Additional key-inputs, namely K_{Dobf}[5], are treated as controllable inputs in the locked netlist.

Although obfuscating the SWC with additional key-gates changes the probability of switch-signals, it does not compromise the SAT resilience of SKG-Lock, as shown in the following proof.

Proof for SAT resilience: We consider the case of an obfuscated switch controller and one SKG controlled by the least-corruptibility switch signal.

In any iteration, with the corresponding DIP X_i, the condition that a wrong key must satisfy is:

$$\begin{aligned}\big[(K_A = 0) \wedge (K_{D_{obf}} = 0) \wedge (\vec{K_D} = \vec{X_i})\big] \vee \\ \big[(K_A = 0) \wedge (K_{D_{obf}} = 1) \wedge (\vec{K_D}[0 : n-2] \neq \vec{X_i}[0 : n-2]) \wedge \\ (\vec{K_D}[n-1] = \vec{X_i}[n-1])\big] \end{aligned} \quad (4.4)$$

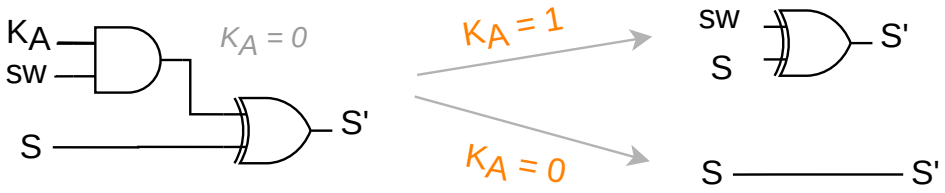
Therefore, when $K_{D_{obf}} = 0$, the wrong key class identified by X_i has the following form:

$$(K_A = 0, K_{D_{obf}} = 0, \vec{K_D} = \vec{X_i}) \quad (4.5)$$

Since there is a one-to-one matching between K_D and X_i, any X_i value can identify a unique set of wrong keys in the form of (4.5). Therefore, every input pattern is a DIP and $N = 2^n$. The circuit is n-secure against SAT attack. Similar to the proof in Section 4.4.3, this proof holds in the case when more SKGs are used.

Thanks to the insertion of multiple SKGs, there is structural entanglement between the SWC and the locked circuit. Hence, the attack based on partitioning algorithm [17] cannot be used to separate this block from the circuit.

[5] K_{Dobf} key-inputs are decoy key-inputs because they have no effect on the circuit's function if K_A is correct.



(a) Vulnerability of SKG-Lock to SCOPE attack

(b) SCOPE countermeasure based on new SKG structure

FIGURE 4.7: Vulnerability and countermeasure of SKG-Lock to SCOPE attack.

SCOPE Attack

The SCOPE attack [41] tackles each key bit individually. For each key-input, it consists in making two circuit copies, each with logic 1 or 0 assigned to the key-input, then synthesizing and optimizing the two circuits, before comparing them using statistical analysis. The deduced key bit is the one associated with the more optimized circuit copy.

In SKG-Lock, there is a structural change when assigning correct or incorrect key bit to an SKG, as depicted in Fig. 4.7(a). If the key bit is correct, the SKG is removed and its switch-signal is shorted, which also lead to further logic reduction. Whereas if it is incorrect, only the AND gate in SKG is removed. Therefore, the SCOPE attack can analyze this structural difference to deduce the correct key.

We propose a countermeasure based on a modified SKG structure. As shown in Fig. 4.7(b), an additional XOR/XNOR gate enables the SKG to be controlled by two key-inputs, one for a K_A bit and the other for a K_D bit. Thus, assigning either value to a key-input only removes the additional XOR/XNOR gate. Therefore, it is challenging for the attack to determine with circuit copy is more optimized, hence, making it unable to recover the key bit.

The following proof shows that the proposed countermeasure does not impact the SAT resilience of SKG-Lock.

Proof for SAT resilience: Wrong key values that can be ruled out by a DIP X_i satisfying the following condition:

$$(K_A = K_D[S]) \wedge (\vec{K_D} = \vec{X_i}) \tag{4.6}$$

For any given X_i, to satisfy the condition in Equation 4.6, there is only one way to select the vector (K_A, K_D)

$$K_A = K_D[S] = X_i[S]$$
$$K_D[j] = X_i[j] \quad \forall j \neq S$$

Thus, each iteration identifies only one wrong key value. Hence, the number of iterations required by the SAT attack to eliminate all (2^n) wrong key values is $N = 2^n$. The circuit is n-secure against SAT attack.

4.5 Experimental Results

We implemented SKG-Lock on ISCAS'85 and MCNC benchmarks. We set in each benchmark an equal size of K_A and K_D, $m = n$ (hence the total key size is $2n$). All n switch-signals were used, each of which was driving each SKG. The experiments were executed on an 8-core Intel processor running at 1.90GHz with 16 GB RAM. ModelSim was used for simulation and measuring output corruption. Synopsys Design Compiler, with a 65nm technology library, was used for estimating overheads.

4.5.1 Security Evaluation

We validated the security proof mentioned in Section 4.4.3 against SAT attack, AppSAT attack and SCOPE attack. The evaluated benchmarks were implemented with SKG-Lock with FLL insertion strategy.

SAT Attack

We first show the evaluation of SKG-Lock against the SAT attack [16] (using the provided tool) to validate the proof in Section 4.4.3.

The evaluation of SAT resilience of SKG-Lock with increasing key size[6](by increasing n) is shown in Fig. 4.8. The expected number of iterations for each case is 2^n in order to be n-secure against the SAT attack. The observed numbers of SAT iterations[7]are bigger than the expected numbers.

We further investigated the relation between SAT resilience and output corruption. In order to create configurations of SKG-Lock with better SAT resistance, we increased the number of SKGs mapped with the lowest corruptibility switch-signal sw_{n-1}, hence restricted the usage of other switch-signals according to the decreasing corruptibility order. In Fig. 4.9 and Fig. 4.10, the number of SKGs mapped with sw_{n-1} ranges from 1 to $n = 10$, where

[6]To better observe the trends, we experiment with small key sizes.
[7]The SAT computation time is proportional to the number of iterations.

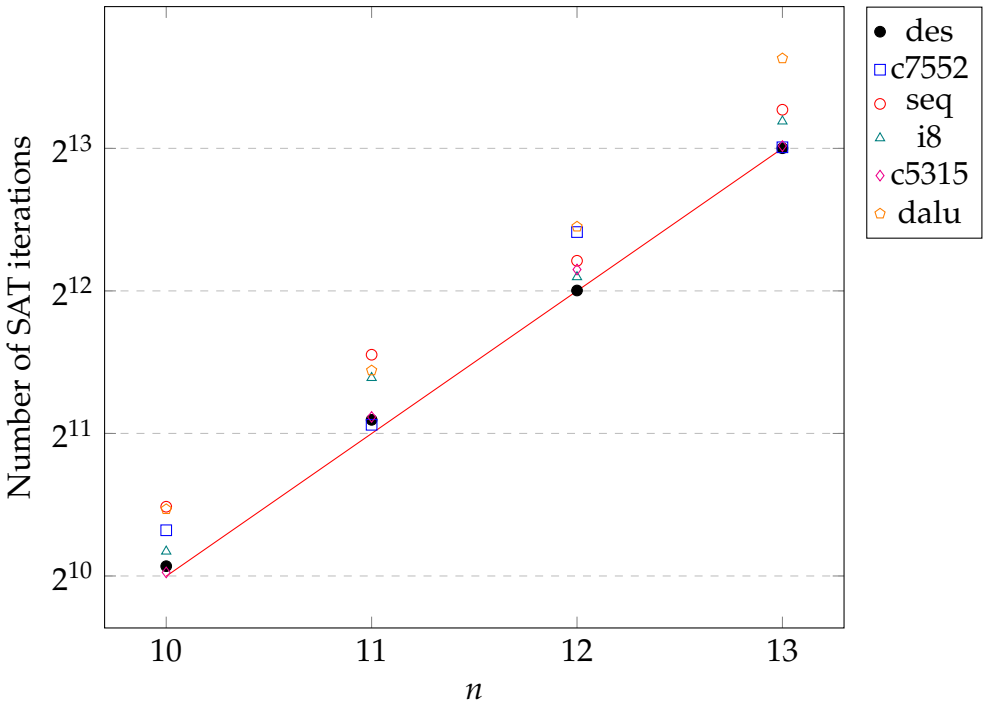


FIGURE 4.8: Evaluation of SAT resilience vs. key size of SKG-Lock.

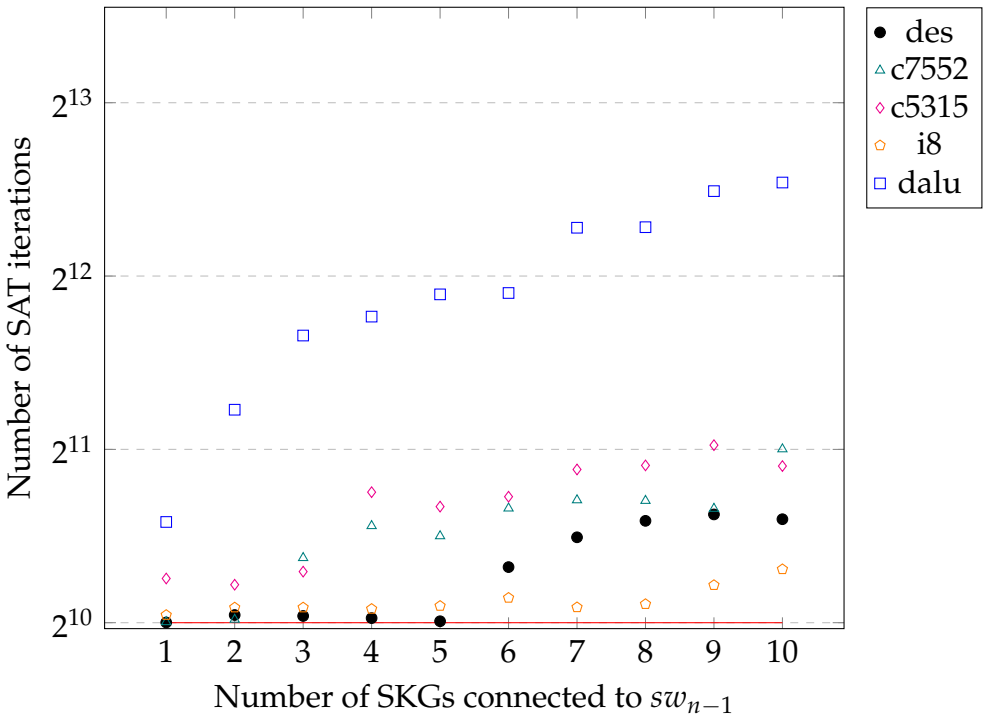


FIGURE 4.9: SAT resilience of SKG-Lock when increasing the number of SKGs connected to the least corruptibility switch-signal ($n = 10$).

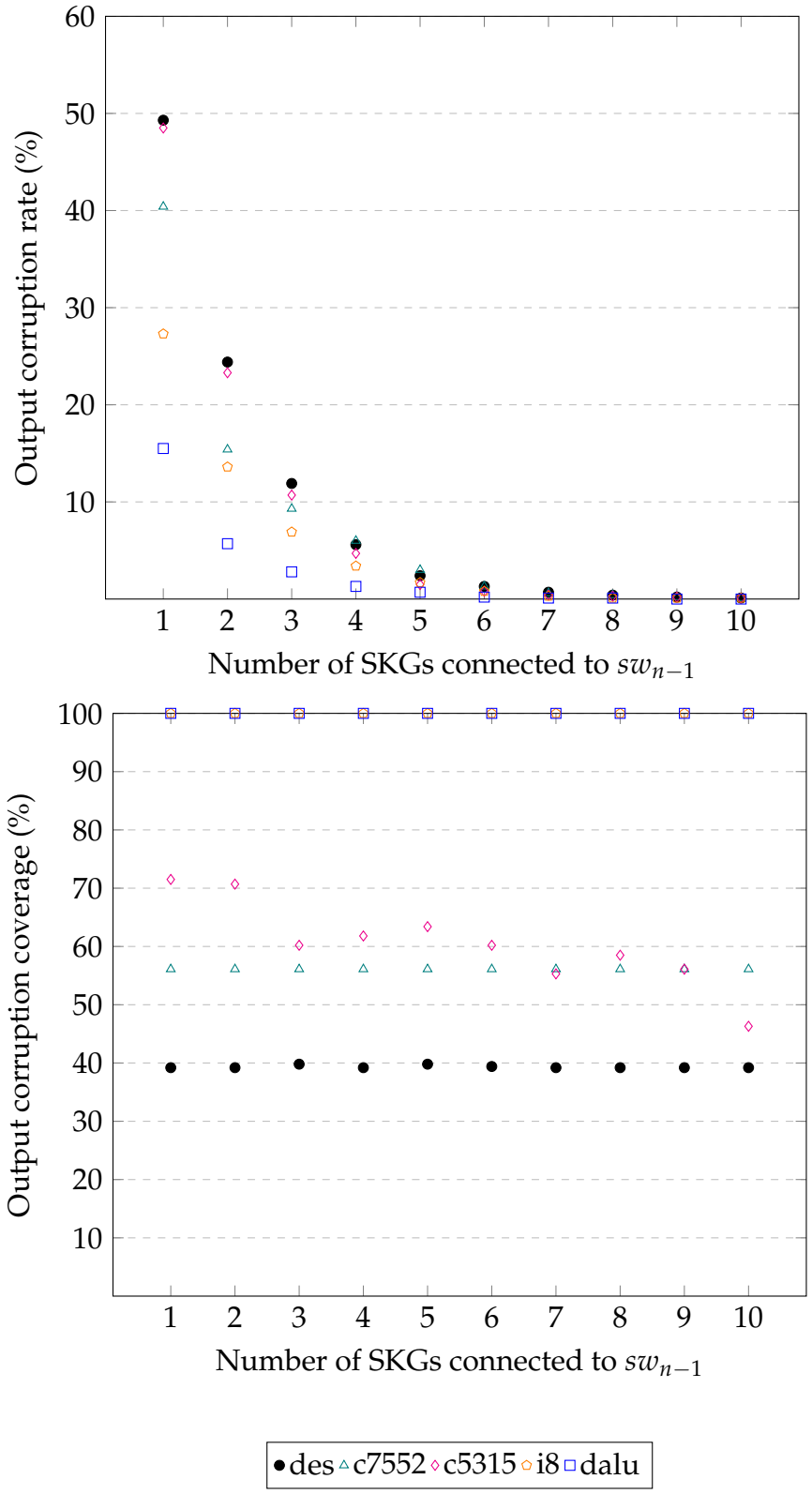


FIGURE 4.10: Output corruption of SKG-Lock circuits in Fig. 4.9.

1 stands for the base configuration — all switch-signals are used and each is mapped to an SKG — and 10 stands for the point-function configuration — only sw_{n-1} is used for all SKGs. The values between the two extremes correspond to the number of SKGs mapped with sw_{n-1}: 9 stands for 9 SKGs mapped with sw_{n-1} and 1 SKG mapped with sw_{n-2}, 8 stands for 8 SKGs mapped with sw_{n-1}, 1 SKG mapped with sw_{n-2} and 1 SKG mapped with sw_{n-3}, etc...

Fig. 4.10 shows that configurations with lower output corruption rate of SKG-Lock can be created by increasing the number of SKGs connected to sw_{n-1}. The result in Fig. 4.9 shows that each obtained number of SAT iterations is higher than the expected 10-secure. A remarkable point is that lower output-corruption configurations are inclined to have higher gain in iterations. Several of the evaluated circuits achieve 11-secure and 12-secure. The cause of the extra iterations could stem from the locations of inserted SKGs. From the proofs in Section 4.4.3, the bits in a DIP that differentiate it from another belong to the part of circuit inputs connected to the SWC. However, propagating SKGs corruption to circuit outputs involves controlling several circuit inputs, not only the ones connected to the SWC. Moreover, several SKGs controlled by the same switch-signal corrupts signals for the same input patterns; hence, there could be masking or interference among corruptions from SKGs. Thus, inputs that are not connected to the SWC may also be taken into account (in addition to the connected ones) when the attack identifies DIPs. Hence, there is a considerable gain in SAT iterations.

Nevertheless, using multiple times sw_{n-1} does not necessarily lead to a significant increase in the number of iteration (between 2^n and 2^{n+1} for most benchmarks), but it significantly reduces output corruption rate (cf. Fig. 4.10). These results confirm that mapping each switch-signal to a different SKG (cf. first column in Fig. 4.9 and Fig. 4.10) is a more optimal way to achieve both high SAT resilience and high output corruption.

TABLE 4.2: APPSAT ATTACK RESULT ON SKG-LOCK ($n = 32$)

Bench	**Accuracy** (%)	**Output Corruption of Key**		
		#Patterns /1000000	Rate (%)	Coverage (%)
des	92.18	1527	0.153	4.8
c7552	86.56	7274	0.73	18.6
i8	81.56	1025	0.103	71.5
c5315	84.37	38683	3.87	28.13
dalu	76.56	9281	0.93	100

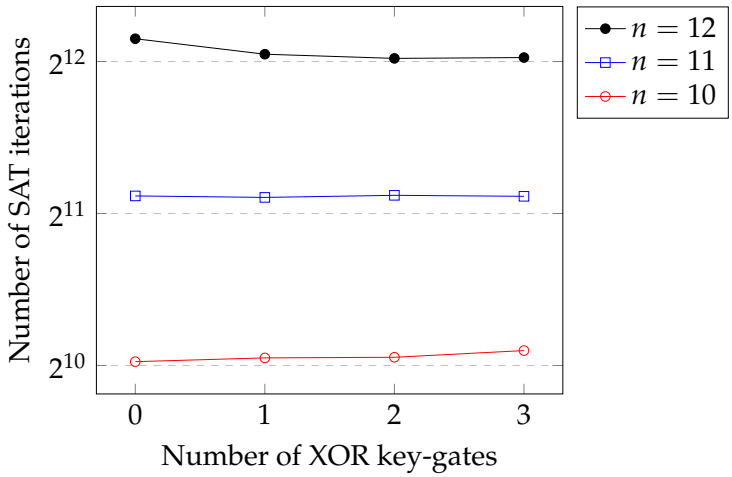


FIGURE 4.11: Evaluation of SAT resilience of SKG-Lock with an obfuscated switch controller (benchmark c5315).

AppSAT Attack

We evaluated SKG-Lock against the AppSAT attack using the NEOS tool [127] from its developer. We used the same attack configuration as in [34]: 50 random queries were applied after every 12 SAT iterations and the settlement threshold was 5. We applied AppSAT on SKG-Lock with $n = 32$ and ran the attack ten times on each benchmark. The accuracy of the approximate keys found by the attack — directly related to the output corruption generated by this key — is of interest.

The result in Table 4.2 shows the average of 10 key values returned from AppSAT. For the accuracy of the key, only the K_A part is taken into account. The column "Accuracy" presents the percent of correct key bits in the resultant keys. For the majority of cases, the accuracy is around 80%, which indicates that the key contains several wrong key bits. Furthermore, we measured the output corruption with the produced keys: for each key value, we applied 1,000,000 random input patterns and compared the outputs observed from the locked circuit to the golden outputs. Columns "#Patterns" (the number of input patterns that produce corrupted outputs), "Rate" (output corruption rate) and "Coverage" (output corruption coverage) in Table 4.2 show the average results of 10 obtained key values. It is apparent that AppSAT failed to reduce the output corruption rate of SKG-Lock to a point-function corruptibility ($1/2^{32}$ in this case). The observed corruption rates range from 0.1% to 4%, indicating that the circuit, if run at 1MHz, may produce several thousands of errors each second. Moreover, we also observed sufficient output corruption coverage on several benchmarks.

TABLE 4.3: SCOPE ATTACK EVALUATION ON SKG-LOCK ($n = 64$)

Bench	**Without countermeasure**		**With countermeasure**	
	Runtime (s)	**Accuracy** (%)	**Runtime** (s)	**Accuracy** (%)
des	10.96	100	11.29	0
c7552	8.44	100	8.6	0
c5315	7.18	100	7.33	0
i8	7.16	100	7.21	0
dalu	6.61	100	6.79	0

TABLE 4.4: OUTPUT CORRUPTION EVALUATION ON SKG-LOCK ($n = 64$)

Bench	**Output Corruption Rate** (%)			**Output Corruption Coverage** (%)		
	SKG-Lock KIP	SKG-Lock FLL	CAS-Lock	SKG-Lock KIP	SKG-Lock FLL	CAS-Lock
des	48.5	49.4	23.47	100	100	0.8
c7552	50	49.6	8.79	61.68	58.88	0.93
c5315	49.5	23.9	12.46	63.42	78.05	1.63
i8	36	11.6	8.89	100	100	1.235
dalu	12.24	31	3.12	100	100	25
Average	**39.25**	**33.1**	11.35	**85.02**	**87.39**	5.59

Removal Attack

We implemented removal countermeasure by obfuscating the SWC with randomly inserted XOR key-gates. We evaluated the SAT resilience of SKG-Lock with obfuscated SWC (cf. Section 4.4.4). As can be seen from the result in Fig. 4.11, the SAT resilience level is maintained even when the number of XOR key-gates used for obfuscation increases.

SCOPE Attack

We validated the security of the SCOPE countermeasure for SKG-Lock proposed in Section 4.4.4. We used the tool provided with the SCOPE attack [41]. The results are reported in Table 4.3. With the original SKG, SCOPE was able to return the correct K_A bit for all SKGs. We implemented SKG-Lock benchmarks using the proposed modified SKGs as SCOPE countermeasure. In this case, the attack returned the value "X" for each key bit since it cannot extract any differences in the circuit when the key bit is 0 or 1. Thus, the proposed countermeasure is secure against the SCOPE attack.

TABLE 4.5: MAXIMUM OUTPUT CORRUPTION OF SKG-LOCK ($n = 64$)

Bench	Output Corruption Rate (%)		Output Corruption Coverage (%)	
	SKG-Lock KIP	SKG-Lock FLL	SKG-Lock KIP	SKG-Lock FLL
des	97	98.61	100	100
c7552	99.9	99.9	64.5	86.9
c5315	99	47.87	68.3	78
i8	71.63	22.16	100	100
dalu	24.35	61.92	100	100

4.5.2 Output Corruption Evaluation

We implemented SKG-Lock benchmarks ($n = 64$) with KIP and FLL insertion strategy and provided a comparison with CAS-Lock for the same SAT resilience level. For CAS-Lock, the CAS-Lock block is inserted at a high-controllability signal in the circuit; the inserted block contains a cascade of AND gates followed by an OR gate, which allows highest corruptibility among all configurations of CAS-Lock.

Table 4.4 presents the results for output corruption rate and output corruption coverage. As can be seen, for both insertion strategies, SKG-Lock achieves better results than CAS-Lock in both metrics. In average, the amounts of output corruption from both strategies are quite equivalent; the KIP strategy provides better output corruption rate compared to FLL strategy. One can notice that, due to the scattering of SKGs throughout the circuit, SKG-Lock is able to affect all circuit outputs in several cases. Conversely, CAS-Lock only corrupts one signal in the circuit, thereby affecting only a few outputs.

In terms of output corruptibility, SKG-Lock also gets significantly better results than CAS-Lock. Furthermore, the KIP strategy provides better results compared to the FLL strategy. For example in *des* benchmark, SKG-Lock with KIP has 16.5% output corruptibility, whereas SKG-Lock with FLL has 1% and CAS-Lock has 0.1%.

In comparison, for the same SAT resilience, SARLock, Anti-SAT and SFLL-HD^0 have a corruption rate of $1/2^{64} = 5.4e^{-18}\%$.

We also measured the maximum output corruption that SKG-Lock can produce. It is the case when all inserted K_A bits are wrong. The results reported in Table 4.5 show that in several cases, the output corruption rate reaches close to 100%. The high output corruption rate is achieved thanks to the SWC structure in Section 4.4.1.

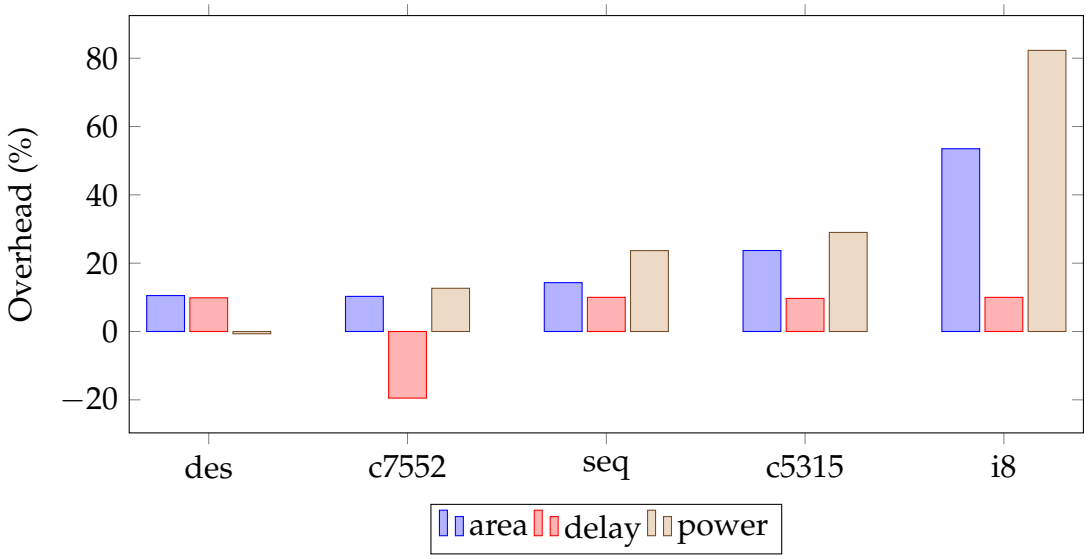


FIGURE 4.12: Overhead evaluation of SKG-Lock ($n = 32$). Benchmarks are in decreasing size order.

4.5.3 Overhead Evaluation

The overhead of SKG-Lock was evaluated in terms of area, power and delay overheads. The benchmarks were implemented with SKG-Lock with KIP insertion strategy.

Fig. 4.12 shows the overhead on different benchmarks for $n = 32$. We set the maximum delay overhead as 10%. For medium benchmarks like des and c7552, the average overhead is less than 10%. For smaller benchmarks, the overhead is bigger.

Fig. 4.13 presents the overhead as the key size increases on benchmark c7552. The area and power overhead scale linearly with the key size. The delay overhead is equivalent to the number of SKGs inserted at the critical path of the circuit. Thus, it does not necessarily increase for a bigger key size.

4.6 Comparison with Related Works

We also provide comparison with other related SAT-resilience techniques, namely SARLock [18], Anti-SAT [94], SFLL-HD [24] and CAS-Lock [26]. Similar to SKG-Lock, these techniques utilize point-function like structures and render the SAT attack exhaustive with exponential iterations.

The comparison is presented in Table 4.6. Whereas SARLock and Anti-SAT are n-secure but with exponentially reduced output corruption rate of $1/2^n$, SFLL-HD enables trading resilience for better corruption rate ($\binom{n}{h}/2^n$). Unlike those techniques, SKG-Lock and CAS-Lock do not conform to the SAT resilience — corruption trade-off. Both offer high SAT resistance; however,

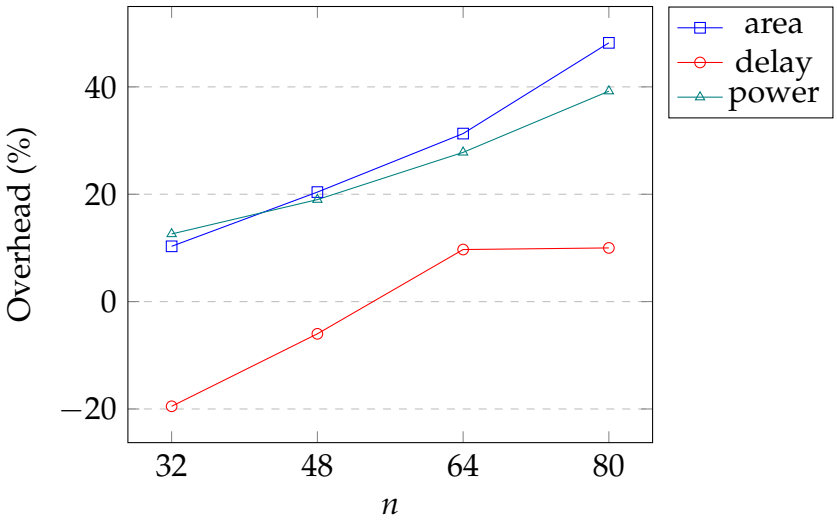


FIGURE 4.13: Overhead vs. key size of SKG-Lock (benchmark c7552).

SKG-Lock provides much higher output corruption (cf. Section 4.5.2). For overhead comparison, we implemented the mentioned techniques in benchmarks used in previous section with $n = 32$ (n is the number of circuit inputs connected to the inserted protection logic). SKG-Lock has slightly higher overheads than SARLock, Anti-SAT and CAS-Lock. This is because its inserted structure contains more gates, and many SKGs may be inserted at the critical path of the circuit, whereas the other techniques insert the keyed block at only one node. SFLL-HD has remarkably much higher overheads than others due to added comparators and Hamming Distance counters.

4.7 Conclusion

In SKG-Lock, there exists at least one SKG mapped to the lowest-corruptibility switch-signal to ensure SAT resilience. However, it is observed that such SKG may not be included in every protected output logic cone. We can indeed imagine an attack consisting in a *hill-climbing* search, attacking in priority each logic cone (e.g. with a SAT solver) with SKGs mapped only with switch-signals with a higher corruptibility. Taking into account this constraint is part of our future work to further improve SKG-Lock.

In this chapter, we proposed a novel SAT-resilient logic locking scheme, SKG-Lock. SKG-Lock architecture is based on switchable key-gates and a switch controller controlled by decoy key-inputs. Thanks to the proposed control of SKGs, our solutions, both SKG-Lock^{lw} and SKG-Lock, are provably secure against the SAT attack. Improved from SKG-Lock^{lw} by taking advantage of multiple SKGs, the proposed SKG-Lock provides tremendously higher output corruption, better structural entanglement and better resilience against

TABLE 4.6: COMPARISON OF SKG-LOCK WITH RELATED WORKS

Technique	**SAT resilience level**	**Output corruption**	**Average overhead** †
SARLock [18]	n	very low	6%
Anti-SAT [94]	n	very low	8%
SFLL-HD^h [24]	$n - \lceil log_2(\binom{n}{h}) \rceil$	low	100%
CAS-Lock [26]	n	medium	9%
SKG-Lock	$\geq n$ *	high	11%

† ISCAS'85 and MCNC benchmarks ($n = 32$)
* Higher level can be achieved with lower-corruption configurations.

SAT-based attacks. Compared to state-of-the-art works, SKG-Lock provides significant output corruption and high attack resilience, while incurring acceptable overhead.

Chapter 5

Scan Controller for Protecting Logic Locking against Oracle-Guided Attacks

Contents

5.1 Introduction

As logic locking is a security primitive in the circuit, it is necessary to design circuit's infrastructure that does not leak the secret data, i.e., the logic locking key. On the offense side, according to the oracle-guided attack model, the attacker is supposed to have the possibility to access to the locked netlist and an unlocked IC used as the oracle. Research on attacks on logic locking has shown that the most powerful attack scenario is when the attacker has the direct input control and output observation of the locked combinational logic in the oracle. This capability is fully granted by the *open* access to the scan chains in the oracle, as illustrated in Fig. 5.1. Indeed, combinational blocks of an IC are generally surrounded by sequential elements (i.e., FFs) for synchronisation. Moreover, these internal FFs can be accessed since original FFs are generally replaced by scan FFs for production test purpose [79]. Scan chains are shift registers formed by linking scan FFs together. A scan chain's input, called Scan-In, is a fully controllable PI and its output, called Scan-Out, is a fully observable PO. Ultimately, by running shift operations in the scan chain, data stored into scan FFs can be read and modified. This allows ATPG tools to model sequential designs as combinational, by assuming internal FFs' outputs as pseudo PIs (PPIs) and their inputs as pseudo POs (PPOs), which decreases the test generation complexity.

Powerful oracle-guided attacks that take advantage of scan chain access have been proposed. Key sensitization attacks [57] identify the correct key bits by analyzing logic cones and sensitizing each key bit to observable outputs. This task requires that the targeted logic cones' inputs are controllable (PIs or scan FFs) and their outputs are observable (POs or scan FFs).

SAT-based attacks [16], [34], [36] exploit scan chains to target locked combinational part in the IC. A preliminary requirement of such attacks is to model the circuit as a Direct Acyclic Graph (DAG). However, combinational circuits can be interpreted as DAGs whereas sequential circuits cannot. Similar to

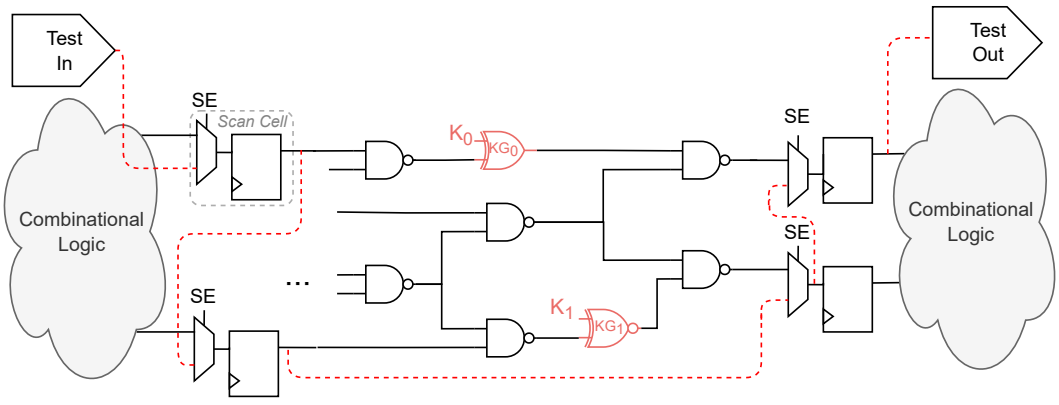


FIGURE 5.1: Scan chains allow control and observation of internal locked combinational logic.

ATPG tools, assuming that internal FFs are replaced by scan FFs, a combinational model is obtained by considering that FFs' outputs and inputs are PPIs and PPOs respectively. With the combinational model of the locked design, in each iteration, the attack finds a DIP, which contains the assignment for both PIs and PPIs of the circuit. In each iteration, the attack finds a DIP to prune out incorrect keys. This DIP is then applied to PIs and PPIs, via scan chain access, of the oracle; and the golden output, including POs and PPOs, is obtained.

As the access to scan chains satisfies the requirement of mentioned attacks, they can be prevented by including an authentication step to block scan shift operations performed by the attackers. This type of solution is suitable for the defender such as the design house who is in charge of DfT insertion during the design process. In this case, the design house can decide to insert a secure scan solution to further protect their logic-locked hardware design. In addition, it can adapt to the modified test procedure involved by the scan protection.

In this chapter, we present a Design-for-Security scan solution that guarantees full testability without compromising the security of logic locking. We propose a scan controller that limits the scan access only to authorized users. We provide suitable logic locking techniques to combine with the proposed solution. A security analysis shows its capability to prevent critical oracle-guided attacks on logic locking. Furthermore, we analyse potential attack schemes on the scan controller and provides comparison with state-of-the-art scan protection methods.

The rest of the chapter is organized as follows. Section 5.2 presents the proposed scan protection technique. The analysis of the solution, including security, testability and overhead analysis, is detailed in Section 5.3. Section 5.4 shows a comparison of our proposal with related works. Finally, Section 5.5 concludes the chapter.

5.2 Proposed Solution

The main idea of the proposed solution is to use a key-based authentication for controlling the activation of scan chains. Upon the insertion of an incorrect value to the scan controller, no scan data is available for attack purposes.

Designers can protect their hardware IPs by applying logic locking and our proposed scan controller. In this case, the design is locked with a *logic locking key*, referred to as K_L; and its scan chains are disabled with a *scan access key*, referred to as K_S. The scan chains can be enabled for testing by providing a required bit stream at the additional *Test Key* port, referred to as K_T.

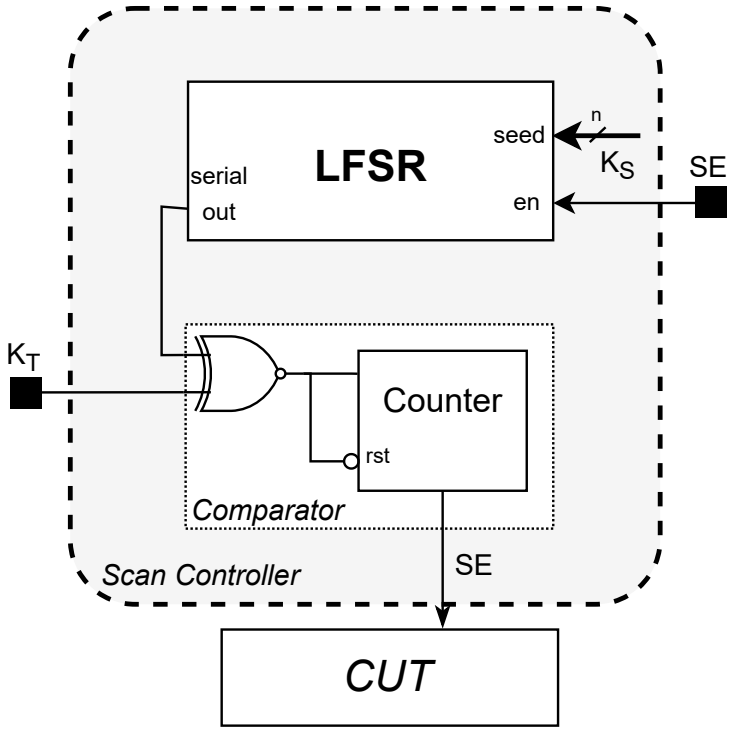


FIGURE 5.2: The structure of the proposed scan controller.

5.2.1 Scan Controller

The structure of the proposed scan chain controller is illustrated in Fig. 5.2. It includes an n-bit Linear-Feedback Shift Register (LFSR) and a comparator. The scan controller "locks" the Scan Enable (SE) signal by setting it to “0". SE is set to “1" only upon the continuous insertion of a correct sequence through the K_T pin. The correct K_T bit stream should match the output of the LFSR. K_S initializes the LFSR and, thus, defines the required K_T value.

The LFSR is an n-bit shift register. It randomizes its state at every clock cycle, i.e. $St_{LFSR_t}(K_S) \neq St_{LFSR_{t+1}}(K_S)$ at any cycle t. As the LFSR is used as a bit stream generator, its output throughout n cycles represents one of its states. The comparator consists of a XNOR gate and a counter. The XNOR gate compares the inserted K_T value and the current output of the LFSR, and feeds the result to the counter. The counter checks if the K_T stream matches the state of the LFSR, i.e. the value of K_T is correct for n *consecutive* cycles. If “1" has been inputted to the counter for the last n cycles, it sets SE signal to “1"; otherwise, SE remains at “0". If the comparison result becomes “0", the counter resets immediately and SE is set to “0". Given the correct K_T bit stream, after the first $n-1$ cycles, SE is set to “1" continuously.

Given the LFSR structure of the scan controller and the K_S value (i.e., its seed), an authorized tester can build an equivalent model of the LFSR to generate the required K_T bit stream for enabling scan access.

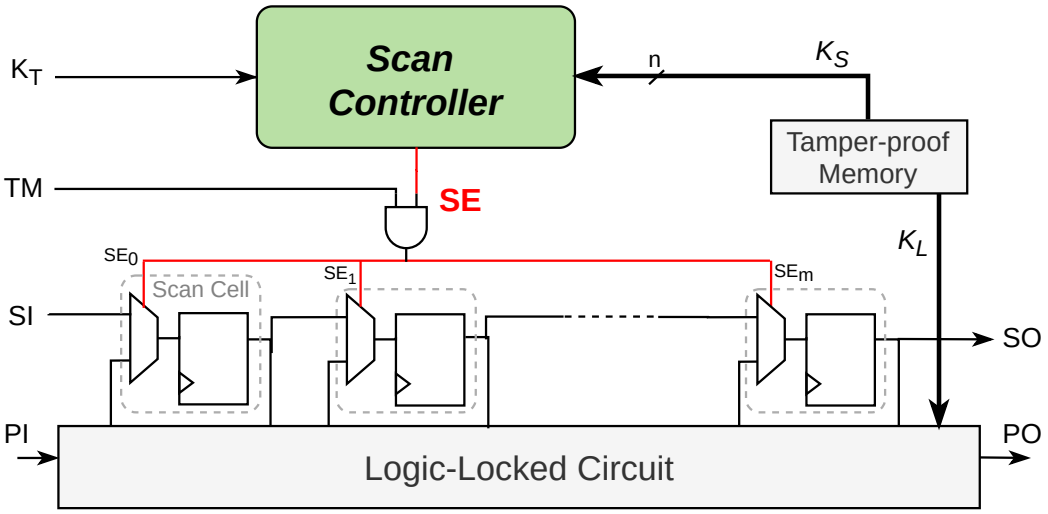


FIGURE 5.3: Proposed logic locking scheme including scan controller and logic locking.

5.2.2 Proposed Logic Locking Scheme

Our proposed logic locking scheme consists of a two-layer protection, the scan controller and the key-gate based logic locking technique using KIP strategy proposed in Chapter 3. In the design flow, it consists in two steps, key-gate insertion in logic locking step and scan controller integration in DfT insertion step. Its architecture, depicted in Fig. 5.3, includes the logic-locked design, a tamper-proof memory and the DfT structure that contains the scan controller.

To switch scan FFs from functional mode to shift mode, a Test Mode (TM) signal is used to set the SE pin in each scan FF to “1". The TM signal also avoids unwanted mode switching due to the periodic nature of the LFSR.

K_L and K_S are stored in the TPM that is programmed by the designer during the activation phase as depicted in Fig. 5.4. It is possible to program each IC with a different value of K_S. In this case, each IC requires a distinct K_T bit stream that can not be reused on another IC. This property can be exploited to help designers track scan usage in each IC.

The design is locked with key-gate insertion using KIP strategy. It is a suitable option since the scan controller provides security against oracle-guided attacks, as analyzed in Section 5.3.1. It provides very high output corruption and costs low hardware overhead (only one XOR/XNOR gate for each key bit).

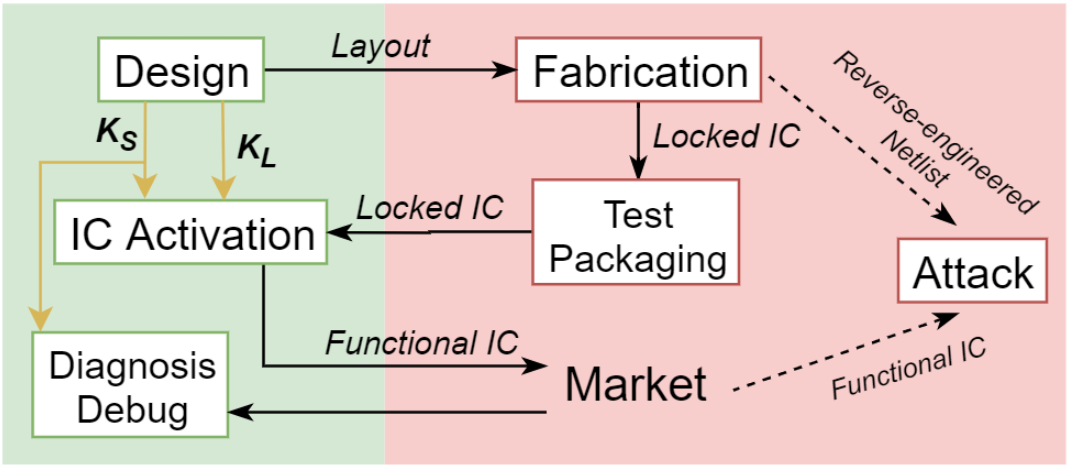


FIGURE 5.4: The production flow with logic locking and scan controller, and the corresponding threat model on logic locking.

5.2.3 Production Flow & Threat Model

Fig. 5.4 depicts the production flow with logic locking and the proposed scan controller. Logic locking and scan controller are implemented during the design process. Logic locking is applied after logic synthesis while the scan controller is inserted during DfT insertion. The foundry, the test/packaging facility and the end user, depicted on a red background in Fig. 5.4, are potential attackers who want to recover K_L. After fabrication and packaging, the ICs are sent to the designer or a trusted entity, who will then program K_L and K_S into their TPM.

For the goal of securing logic locking, potential attackers should not be in possession of K_S. The designer should restrict the distribution of K_S only to trusted partners. The manufacturer is commonly in charge of conducting production test. It includes a scan-based structural test that is performed on wafers by probing each die. Thus, structural test should be performed without the knowledge of K_L and K_S. Structural test can be performed on a logic-locked circuit without K_L. Test patterns for such test are generated assuming that key inputs are controllable. This procedure is shown to provide maximum fault coverage while preventing attackers from retrieving K_L by analysing test patterns [80]. To enable the scan access during manufacturing test without sharing the secret K_S value, a hardcoded value is used as a temporary K_S value during the test and this value can be shared with the tester. To bypass the K_S input coming from the TPM, the designer can insert a multiplexer controlled by a pull-up element or a controllable source. The former is based on the Saw Bow method [128], where the connecting wire crosses the sawing line of the wafer. The latter is depicted in Fig. 5.5. An additional pin is used to control the multiplexer that selects K_S either from the TPM or the hardcoded value. It is blown off after the packaging process; hence, the circuits used in field have K_S coming from the on-chip TPM. In case when

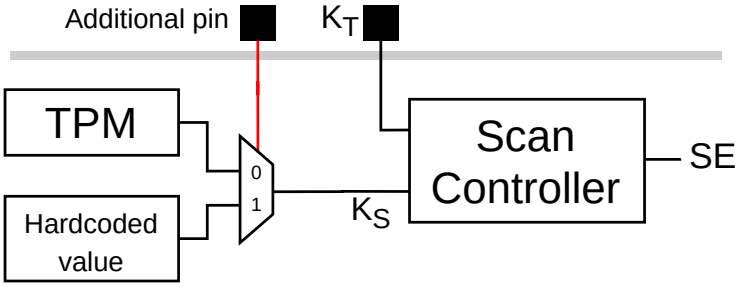


FIGURE 5.5: Using hardcoded value for K_S of the scan controller during manufacturing test.

an IC presents an erroneous behavior in the field and the failure analysis requires the access to scan chains, debug and diagnosis must be conducted by the design house or a trusted partner.

5.3 Results & Analysis

5.3.1 Security Analysis

Attacks on Scan Controller

Since the proposed scan controller is used as a protection for logic locking, attackers on logic locking may aim to break this scan protection prior to targeting logic locking. In this case, as the threat model of logic locking is adopted, the attacker is assumed to have access to a functional chip and a corresponding reverse-engineered netlist. In this section, we analyze anticipated attacks on the scan controller.

K_S Recovery Attacks

K_S is the secret of the scan controller; if the attacker can recover K_S, they can activate the scan chains and attack the implemented logic locking.

As mentioned in section 5.2.1, K_T has to be correct for the last n cycles in order to enable the scan chains for 1 cycle. The probability of guessing this value with a brute-force approach is $1/2^n$. SE is the only output of the scan controller and represents the only 1-bit information observable to the attacker. The function of the scan controller at any cycle t can be modelled as a point function:

$$SE = \begin{cases} 1, & \text{if } K_{Tt\rightarrow t+n} = St_{LFSR_t}(K_S) \\ 0, & \text{otherwise} \end{cases} \quad \forall K_S, K_{Tt\rightarrow t+n} \in \{0,1\}^n \tag{5.1}$$

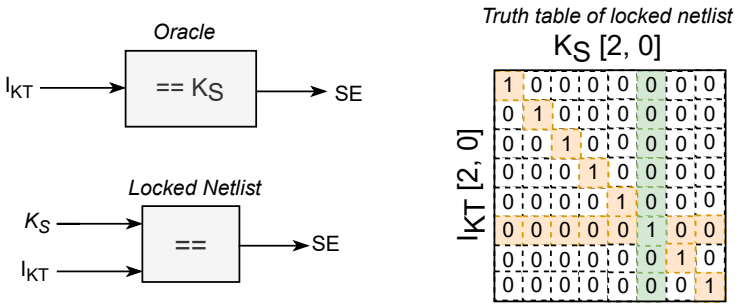


FIGURE 5.6: Applying a SAT-based attack on scan controller to recover K_S.

A point function is a one-way function, which is a fundamental cryptographic primitive and is provably hard to invert. An attacker can resort to cryptanalytic principles to correlate the inserted K_T and the SE signal, observed through scan output. However, due to the point function behavior, it is not possible to derive K_S with a more efficient approach than brute-force guessing.

Point functions have also been proven to be resilient against SAT-based attacks [17], [18]. Using a point function to hide secret information forces the SAT solver to check for every possibility. Thus, applying a SAT-based attack on the scan controller can only eliminate one wrong K_S value each iteration as illustrated in Fig. 5.6, hence reducing the attack's efficiency to that of brute-force. Therefore, using formal methods such as SAT solvers cannot aid the attacker at retrieving the K_S value.

Advanced attacks such as ScanSAT [114] and DynUnlock [115] successfully break recent scan protection proposals [27], [112] (cf. Section 5.4). Using the same attack model as the SAT attack on logic locking, these attacks rely on scan data control and observation from the oracle. However, as mentioned in Section 5.2, data from the scan chains are not available since the scan controller disables the scan chains. Furthermore, signal observed at SO pin shows no correlation with K_S. Therefore, these attacks are not effective against the proposed scan controller.

Tampering Attacks

Certain attackers in the production flow have the capability to tamper the circuit or chip design. An untrusted manufacturer has the capability to modify the mask before IC fabrication, which allows them to bypass or remove the scan controller introduced in the circuit. In another case, an untrusted packaging facility may neglect to destroy the additional pin used during manufacturing test to later use it as backdoor. As the produced ICs are sent back to the design house for activation, the designers can perform an *ad hoc* test

to detect such tampering before inserting K_L. Therefore, the chips with tampered scan controller will not be activated and cannot be used as an oracle for attacks on the implemented logic locking.

Attacks on Logic Locking

The SAT attack needs an oracle with accessible scan chains to control and observe respectively inputs and outputs of the combinational part under attack. The proposed scan controller blocks any unauthorized usage of the scan chains in the oracle. In other words, the attacker cannot apply any DIP generated by the SAT solver to the inputs of the combinational part, and he/she cannot deduce expected outputs for wrong key elimination. Therefore, the attack is unable to solve the problem of finding K_L. The same remarks can be made concerning the sensitization attacks, which also require full control and observation of combinational logic cones.

Sequential SAT-based attacks [37], [107] can attack locked sequential circuits without access to scan chains. They use the time-frame expansion method, a more computationally intensive way to represent a sequential circuit as combinational. The circuit is modelled as a series of copies of its combinational part, where each copy corresponds to a time-frame. The same procedure is used to perform production test pattern generation on sequential circuits without scan chains. However, the number of time-frames must be large enough to reach all the states of the circuit, which can require up to $2^{n_{FF}}$ time-frames, where n_{FF} is the number of FFs. For the same reason, sequential test pattern generation is avoided in practice and scan design is the *de facto* Design-for-Testability approach today. Indeed, the authors report that attacks based on sequential SAT approaches can only handle small circuits with a few thousands gates.

Oracle-less attacks [38], [40], [93] analyze the structure of a locked netlist to guest the key value or to remove inserted logic. In the proposed logic locking scheme, key-gates are inserted for circuit locking. Inserted key-gates and their neighboring gates are structurally transformed during logic synthesis, making them resilient against removal [38]. Key guessing attacks [40], [93] can partially recover the key of inserted key-gates. Such attacks can be hindered with additional constraints [30], [92], [129] to the insertion strategy.

Data Stealing Attacks

The scan controller also improves data confidentiality in the circuit. Scan chains are the target of a plethora of attacks that aim to steal secret data from ICs [84]. The scan controller prevents attackers from using scan structure deliberately. Thus, malicious data cannot enter the scan chains, nor can secret data be examined from the scan chains.

TABLE 5.1: FAULT COVERAGE OF THE SCAN CONTROLLER WITH DIFFERENT SIZES OF LFSR

Scan Controller	**Fault Coverage** (%)
64-bit LFSR	98.89
80-bit LFSR	98.39
128-bit LFSR	99.29

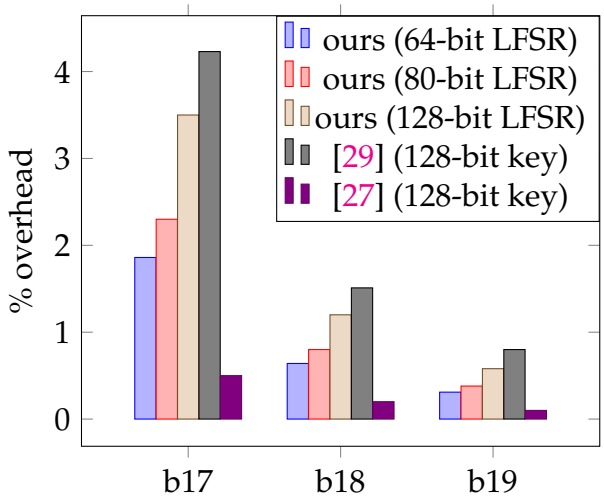


FIGURE 5.7: Area overhead of the proposed scan controller and related works on ITC'99 benchmarks.

5.3.2 Testability Evaluation

Faults in the scan controller can easily be propagated to the SE signal and affect the activation of the scan chains. Thus faults can be observed during the testing of CUT, when the correct K_T sequence is inserted to the scan controller to activate the scan chains. For evaluating the fault coverage of the scan controller, we used fault simulation. We generate K_T bit stream for simulating faults in the scan controller with Synopsys TetraMax. Table 5.1 presents the fault coverage of the scan controller with different sizes of LFSR. A small number of stuck-at-faults in the LFSR were not observed during the simulation. The implementation of the scan controller can be further optimized to make it more testable.

The scan controller does not have any impact on the fault coverage of CUT. As discussed in Section 5.2.3, the CUT, even before being unlocked, can be tested with structural testing by treating the key-inputs as controllable inputs.

5.3.3 Overhead Evaluation

We implemented the scan controller in ITC'99 benchmarks [130]. The benchmarks were synthesized on a 65nm technology library with Synopsys Design

Compiler. Fig. 5.7 shows the area overhead for three versions of the proposed scan controller using a 64-bit, an 80-bit and a 128-bit LFSR, along with a comparison with related works [27], [29] using a 128-bit key. The area overhead of the scan controller is as low as 0.3% in the b19 benchmark. Although the solution in [27] has a smaller area cost, we show in the next section that it is less secure than the proposed solution.

We also evaluated the test time overhead of the scan controller. We used Synopsys TetraMax to generate test patterns for the mentioned benchmarks. The b18 benchmark is estimated to have a test time of about $6 \cdot 10^7$ clock cycles. Using a 128-bit scan controller takes 127 initial cycles to set up the SE signal. Thus, it represents a largely trivial test time overhead.

5.4 Comparison with Related Works

Research on secure scan chain design has been established since the introduction of scan-based side channel attacks on crypto circuits [84]. The attacks deduce the cipher key by analyzing the partial results of the cryptographic operation that can be observed through the scan chains. However, countermeasures against such attacks may not adapt to the threat model of attacks on logic locking, where the attacker has the additional capability of reverse-engineering the circuit. Solutions that aim at masking scan data and assume that the scan chain structure is unknown to the attacker, such as [131] which judiciously inserts inverters into the scan chains, are therefore vulnerable in the scenario of logic locking. Other solutions based on manipulating scan data during the switch between functional mode and test mode [132] also turn nullified as SAT attack or ScanSAT attack do not need to switch the circuit to functional mode. Therefore, recent works have proposed secure scan solutions that accommodate the stronger threat of attacks on logic locking [110].

Recent scan-based defenses for logic locking include scan locking [27], [28], [111], [112] and scan blockage [29]–[31], [116], [117].

Scan locking [27], [111], [112] consists in inserting key-gates at scan path in order to corrupt scan data. However, the locked scan chains can be treated as a combination of key-gates inserted at PPIs and PPOs of the circuit. With this combinational model, SAT-based attacks such as ScanSAT attack [114] and DynUnlock attack [115] are able to find the scan access key of such techniques. A recent technique, Dynamically Obfuscated Scan Chain [28], is shown to be resistant against SAT-based attacks that recover the scan access key. However, the solution waits for several cycles before corrupting scan data, i.e., switching the key-gates inserted at scan chains. Therefore, correct scan data is still available for implementing oracle-guided attacks on logic locking [30].

Scan blockage [29], [116] is based on stitching key-registers into scan chains, in order to change the content of key-registers when the scan chains are activated. However, due to testing support, the scan shift operation is still available [29], [116]. The key bits can be shifted to other scan cells and then be sensitized to observable POs [31], [116]. Techniques in [30], [31] mitigate such vulnerability with a dedicated scan chain for key-registers. Nevertheless, since key-registers are not in close proximity, connecting them into one scan chain can lead to place-and-route issue in layout design. Furthermore, optical probing can extract key from such key-registers [133].

Scan controller freezes the scan chains for unauthorized users. Thus, scan data is not available and scan shift operation cannot be used. Therefore, attacks on mentioned scan techniques cannot be applied to scan controller. Furthermore, compared to such techniques, scan controller has a small overhead.

5.5 Conclusion

The scalability and efficiency of powerful oracle-guided attacks on logic locking rely heavily on the control and observation of inputs/outputs of the combinational part under attack. While scan chains are widely used in the industry for production test purpose, they also provide such capability and, thus, make these attacks possible.

In this chapter, we present a low-cost scan controller that requires a key sequence to enable the shift operations in the scan chains. The resulting blocked scan access prevents adversaries from shifting data in or out from the scan chains, making it impossible to implement the attacks. Using the scan controller with key-gate insertion techniques results in a logic locking scheme that is highly effective and secure against numerous attacks. Security analysis shows its ability to secure the scan access key and its robustness against tampering. Full testing is supported when performed by authorized partners without leaking secret data to adversaries. In addition, the solution is scalable and easy to integrate.

Chapter 6

Conclusion & Perspective

Contents

6.1 Conclusion

IC overproduction, IP piracy and hardware Trojans are well recognized security concerns in the globalized semiconductor supply chain, where increasing outsourcing to external contractors leads to possible exposure of hardware design to adversaries. Their consequences lead to financial loss and, more dangerously, national security threats. Therefore, a holistic Design-for-Trust method like logic locking is of great interest. Combinational logic locking consists in inserting key-controlled logic in combinational netlists so that the circuit does not function correctly unless the key is provided. Thus, it allows hardware design owners to gain control over their products throughout the production. Thanks to its broad threat model and active protection, it can address aforementioned threats: it can be used at IP core level to hinder IP overuse, potentially prevent IP reverse engineering and HT insertion, and at system level to thwart IC overproduction. For these purposes, the requirements for logic locking are to secure the logic locking key as well as effectively disrupting the functionality of the locked circuit. Ensuring functionality disruption necessitates sufficient output corruption, which is influenced by the insertion strategy of key-controlled logic. For security evaluation of logic locking, numerous attacks have been developed. The SAT attack introduced the most effective attack method, which is based on Boolean Satisfiability and is oracle-guided. Using a SAT solver allows quickly distinguishing the correct key from the incorrect ones so that each iteration of the attack can eliminate a large number of wrong keys. In addition, the attack takes advantages of scan chains in the oracle to directly target each logic-locked combinational part, which makes the attack computationally feasible. Consequently, strong resistance against the SAT attack has been a priority for secure logic locking techniques. Existing techniques based on point-functions are provably SAT-attack secure, however, at a cost of minimal output corruption. Besides, not only the SAT attack, but other oracle-guided attacks also exploit the scan chains of the oracle. In this thesis, we identify three aspects for developing secure and effective logic locking, which are insertion strategy, SAT-secure logic lock and scan chain protection. The contributions of the thesis are three folds as follows.

IN CHAPTER 3, we proposed KIP, a key-gate insertion strategy optimized for output corruption. For the strategy, nodes in the circuit are ranked according to their output corruption score, where a high score means that corruption at such location is highly observable at multiple outputs. Based on signal probability analysis, the score of a node is calculated by measuring the change in the probability of outputs if corruption happens at such node. After the ranking, nodes with highest scores are selected for key-gate insertion. This ranking scheme does not require repeated score calculation like FLL, which makes KIP more scalable than FLL. In addition, this metric can be used to prevent inserting key-gates in series. We demonstrated the proposed strategy for

XOR/XNOR key-gate insertion and compared with FLL, the most optimized solution for output corruption. KIP achieves optimal results in all output corruption metrics, which is is as effective as FLL, while taking much less execution time to lock a circuit.

IN CHAPTER 4, we proposed SKG-Lock, a SAT-secure and effective logic locking technique [118]. Two fundamental components are switchable key-gates controlled by activation key-inputs and switch controller controlled by decoy key-inputs. We proved that SKG-Lock^{lw}, a light-weight option of SKG-Lock, achieves the same SAT resilience level while requiring half the overhead compared to point-function based logic locking. The proposed SKG-Lock framework allows insertion of multiple SKGs that have different corruptibilities. It provides significant output corruption and structural entanglement while achieving provable SAT resistance. Its output corruption is optimized by using the KIP strategy for SKG insertion and improving the SWC structure. We provided security analysis of SKG-Lock with SAT-based attacks, as well as other effective oracle-guided and oracle-less attacks. Countermeasures against the SCOPE attack and the probability-analysis based removal attack were proposed. Compared to related SAT-resilient techniques, SKG-Lock provides significantly higher output corruption and better resilience against SAT-based attacks. Overhead evaluation shows that SKG-Lock incurs acceptable overhead on small benchmarks.

IN CHAPTER 5, we proposed a secure scan controller for protecting logic locking against oracle-guided attacks [134]. The scan controller consists in a key-based authentication mechanism that controls the *Scan Enable* signal of the scan chains once the IC is deployed in the field, i.e., the oracle. The scan chains are enabled only only upon the continuous insertion of a correct sequence through the Test Key port. The solution freezes the scan chains for unauthorized users, which prevents implementing powerful oracle-guided attacks on logic locking including SAT-based attacks. The resulting logic locking scheme, composed of two-layer protection based on the scan controller and the key-gate insertion technique using KIP strategy, is highly effective and secure against numerous attacks. Adapting from the threat model of logic locking, we identified potential attack schemes on the scan controller. Thanks to the comparator-based mechanism, attacks that aims at guessing the scan access key cannot be more efficient than brute-force. Potential removal and bypass of the scan controller can be detected. The solution is testable, easy to integrate and supports full testing. In addition, it has low overhead compared to related scan-based defenses.

Ultimately, the contributions in this thesis construct two logic locking schemes:

(i) **SKG-Lock + KIP**: SKG-Lock using KIP strategy for SKG insertion;

(ii) **KIP + Scan Controller**: a two layer protection, XOR key-gate insertion with KIP strategy for circuit locking and scan controller for scan protection.

SKG-Lock + KIP It is a generic logic locking technique that can be used in any case by any defender. It provides provable security against the SAT attack as well as high resilience against other efficient oracle-guided and oracle-less attacks. However, the use of a point-function structure can leave structural vulnerability. It shows significant results in both output corruption rate and coverage. The overhead is reasonable for IP core.

KIP + Scan Controller It presents a two-step protection, key-gate insertion in logic locking step and scan controller integration in DfT insertion step. Thus, the scheme is suitable in cases when manipulations in DfT insertion and test generation are allowed. Comprehensive protection against oracle-guided attacks is achieved. However, the scheme may still be vulnerable to potential oracle-less attacks. Key-gate technique using KIP strategy provides very high output corruption. The overhead from inserted key-gates in protected core is small. The scan controller adds acceptable hardware overhead to the DfT infrastructure of the circuit. It also incurs overhead in test time.

Each scheme presents a different direction towards secure and effective logic locking, with different advantages. The following subsection presents future developments for both schemes.

6.1.1 Future Works

The contributions of this thesis can be further improved and extended.

SKG-Lock The KIP strategy used for SKG insertion is optimized for output corruption. However, it does not ensure that every protected logic cone contains the SKG controlled by the least corruptibility switch-signal. This may lead to a vulnerability where an attack prioritizes attacking cones with only SKGs mapped with high-corruptibility switch-signals. The SKG insertion strategy can be improved with this additional constraint. Another vulnerability is the possibility of identifying the switch controller, which stems from its recognizable structure. Detecting the switch controller gives lead to the attacker to identify switch-signals or distinguish decoy key-inputs, which can be exploited to bypass or make the solution more vulnerable to attacks. Strong structural obfuscation method for SKG-Lock is needed for more comprehensive security.

KIP Strategy & Scan Controller The KIP strategy calculates score and rank all the nodes in the netlist. Certain parts of the circuit, for example the controller, have higher impact on the circuit function than others; in other words,

nodes in these regions of the netlist potentially have a high output corruption score. The strategy can focus solely on such regions to optimize its execution time. Although the scan controller provides comprehensive protection against oracle-guided attacks, inserted XOR key-gates can be vulnerable to oracle-less attacks. The KIP strategy can be improved with additional constraints [30], [129] as countermeasures for such attacks. The scan controller can be integrated into the JTAG infrastructure. It presents an additional function that requires authentication to access debug interface of any logic-locked core.

6.2 Future Perspectives

A future direction for logic locking is locking methods for RTL designs. The majority of logic locking techniques are for gate-level netlists. However, these techniques have predictable inserted logic, such as key-gates or point-function locks. As existing synthesis tools are security-agnostic, even after resynthesis of the locked netlist, these additional logic gates/blocks may still be intact. Netlist-analysis attacks based on machine learning [93], [135] have been proposed to exploit such structural vulnerability. Indeed logic locking methods applied at higher abstraction level are potential solutions. Logic locking at RTL [136], [137] transforms an original RTL description to an RTL description protected with key-controlled components. The subsequent logic synthesis and optimization steps can dissolve the traces of such components. Moreover, the synthesis tool also optimizes the design complexity and enables resource sharing.

The RTL design presents semantic information such as operations, control flows and constants. Therefore, RTL locking can be application-oriented, where it protects the IP that is intrinsic to the circuit. For example, in constant multiplication, which is commonly used in neural network accelerators and filters, the constants are the IP to be protected from IP thief via reverse engineering. The technique proposed in [138] presents an RTL locking method for such purpose. It hides the protected constants among additional decoy constants with key-controlled selectors. Since it corrupts the operand values of the multiplier, the technique provides high output corruption. Moreover, the key-inputs are part of the locked multiplier's inputs, which makes the technique resilient against the SAT attack.

Another future direction is to integrate logic locking into EDA solution. Logic locking has received significant interest from EDA companies [68]–[70]. It presents a design methodology with defined security metrics. Including logic locking into EDA flow provides an automated framework that optimizes security level as well as power, performance and area (PPA) of the circuit. Recent works have proposed such solutions [139], [140]. The work in [139] uses a concurrent tree search method or a machine learning approach to estimate the PPA cost to then devise the optimal locking strategy. Another

solution in [140] introduced a two-level optimization method based on simulated annealing and mixed integer linear programming to satisfy security requirements while minimizing PPA cost.

Chapter 7

Scientific Contributions

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

Publications in Proceedings of International Conferences

1. **Quang-Linh Nguyen**, Marie-Lise Flottes, Sophie Dupuis, and Bruno Rouzeyre. "*On Preventing SAT Attack with Decoy Key-Inputs*." In 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 114–19, 2021. https://doi.org/10.1109/ISVLSI51109.2021.00031.
2. Levent Aksoy, **Quang-Linh Nguyen**, Felipe Almeida, Jaan Raik, Marie-Lise Flottes, Sophie Dupuis, and Samuel Pagliarini. "*High-Level Intellectual Property Obfuscation via Decoy Constants*." In 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS), 1–7, 2021. https://doi.org/10.1109/IOLTS52814.2021.9486714.
3. **Quang-Linh Nguyen**, Emanuele Valea, Marie-Lise Flottes, Sophie Dupuis, and Bruno Rouzeyre. "*A Secure Scan Controller for Protecting Logic Locking*." In 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), 1–6, 2020. https://doi.org/10.1109/IOLTS50870.2020.9159730.

Publications in National Conferences without Proceedings

1. **Quang-Linh Nguyen**, Sophie Dupuis, Marie-Lise Flottes, and Bruno Rouzeyre. "*A New Key-Gate Insertion Strategy for Logic Locking with High Output Corruption* ." In Toulouse Hacking Convention, France, 2022.
2. **Quang-Linh Nguyen**, Sophie Dupuis, Marie-Lise Flottes, and Bruno Rouzeyre. "*Design-for-Hardware-Trust*." In Colloque du GDR SoC2, 2019.

Bibliography

[1] M. Rostami, F. Koushanfar, and R. Karri, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2335155.

[2] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014, ISSN: 1558-2256. DOI: 10.1109/JPROC.2014.2332291.

[3] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware Trojan Attacks: Threat Analysis and Countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2334493.

[4] "Xilinx Sues Flextronics Alleging Fraudulent Chip Resale," *Bloomberg.com*, [Online]. Available: https://www.bloomberg.com/news/articles/2013-12-11/xilinx-sues-flextronics-alleging-fraudulent-chip-resale (visited on 11/18/2021).

[5] (). "Ford accused by software maker of intellectual property theft," Automotive News, [Online]. Available: https://www.autonews.com/article/20150604/OEM06/150609919/ford-accused-by-software-maker-of-intellectual-property-theft (visited on 11/18/2021).

[6] "China Used a Tiny Chip in a Hack That Infiltrated U.S. Companies," *Bloomberg.com*, [Online]. Available: https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies (visited on 03/18/2019).

[7] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining Trust in VLSI Design: Design-for-Trust Techniques," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, Aug. 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2332154.

[8] S. Dupuis, M. Flottes, G. D. Natale, and B. Rouzeyre, "Protection Against Hardware Trojans With Logic Testing: Proposed Solutions and Challenges Ahead," *IEEE Design Test*, vol. 35, no. 2, pp. 73–90, Apr. 2018, ISSN: 2168-2356. DOI: 10.1109/MDAT.2017.2766170.

[9] C.-H. Chang, M. Potkonjak, and L. Zhang, "Hardware IP Watermarking and Fingerprinting," in *Secure System Design and Trustable Computing*, C.-H. Chang and M. Potkonjak, Eds., Cham: Springer International Publishing, 2016, pp. 329–368, ISBN: 978-3-319-14971-4. DOI: 10.1007/978-3-319-14971-4_10. [Online]. Available: https://doi.org/10.1007/978-3-319-14971-4_10 (visited on 11/04/2021).

[10] F. Koushanfar, "Hardware Metering: A Survey," in *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds., New York, NY: Springer, 2012, pp. 103–122, ISBN: 978-1-4419-8080-9. DOI: 10.1007/978-1-4419-8080-9_5. [Online]. Available: https://doi.org/10.1007/978-1-4419-8080-9_5 (visited on 11/04/2021).

[11] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS '13*, Berlin, Germany: ACM Press, 2013, pp. 709–720, ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516656. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2508859.2516656 (visited on 03/04/2020).

[12] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably Secure Camouflaging Strategy for IC Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1399–1412, Aug. 2019, ISSN: 0278-0070. DOI: 10.1109/TCAD.2017.2750088.

[13] S. Dupuis and M.-L. Flottes, "Logic Locking: A Survey of Proposed Methods and Evaluation Metrics," *Journal of Electronic Testing*, vol. 35, no. 3, pp. 273–291, Jun. 2019, ISSN: 0923-8174, 1573-0727. DOI: 10.1007/s10836-019-05800-4. [Online]. Available: http://link.springer.com/10.1007/s10836-019-05800-4 (visited on 04/16/2021).

[14] A. Chakraborty, N. G. Jayasankaran, Y. Liu, J. Rajendran, O. Sinanoglu, A. Srivastava, Y. Xie, M. Yasin, and M. Zuzak, "Keynote: A Disquisition on Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019, ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2019.2944586.

[15] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "IP Protection and Supply Chain Security Through Logic Obfuscation: A Systematic Overview," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 24, no. 6, 65:1–65:36, Sep. 2019, ISSN: 1084-4309. DOI: 10.1145/3342099. [Online]. Available: http://doi.acm.org/10.1145/3342099 (visited on 10/01/2019).

[16] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, IEEE, May 2015, pp. 137–143, ISBN: 978-1-4673-7421-7. DOI: 10.1109/HST.2015.7140252. [Online]. Available: http://ieeexplore.ieee.org/document/7140252/ (visited on 04/11/2019).

[17] Y. Xie and A. Srivastava, "Mitigating SAT Attack on Logic Locking," in *Cryptographic Hardware and Embedded Systems – CHES*, vol. 9813, 2016, pp. 127–146, ISBN: 978-3-662-53139-6 978-3-662-53140-2. DOI: 10.1007/978-3-662-53140-2_7. [Online]. Available: http://link.springer.com/10.1007/978-3-662-53140-2_7 (visited on 01/23/2020).

[18] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2016, pp. 236–241. DOI: 10.1109/HST.2016.7495588.

[19] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Logic encryption: A fault analysis perspective," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2012, pp. 953–958. DOI: 10.1109/DATE.2012.6176634.

[20] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault Analysis-Based Logic Encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, Feb. 2015, ISSN: 0018-9340. DOI: 10.1109/TC.2013.193.

[21] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On Improving the Security of Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, Sep. 2016, ISSN: 0278-0070. DOI: 10.1109/TCAD.2015.2511144.

[22] A. Saha, S. Saha, S. Chowdhury, D. Mukhopadhyay, and B. B. Bhattacharya, "LoPher: SAT-Hardened Logic Embedding on Block Ciphers," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6. DOI: 10.1109/DAC18072.2020.9218600.

[23] M. Yasin, A. Sengupta, B. C. Schafer, Y. Makris, O. Sinanoglu, and J. J. Rajendran, "What to Lock?: Functional and Parametric Locking," in *Proceedings of the on Great Lakes Symposium on VLSI 2017 - GLSVLSI '17*, Banff, Alberta, Canada: ACM Press, 2017, pp. 351–356, ISBN: 978-1-4503-4972-7. DOI: 10.1145/3060403.3060492. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3060403.3060492 (visited on 04/26/2019).

[24] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-Secure Logic Locking: From Theory To Practice," in *ACM SIGSAC Conference on Computer and Communications Security - CCS '17*, ACM Press, 2017, pp. 1601–1618, ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3133985. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3133956.3133985 (visited on 04/23/2019).

[25] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "ATPG-based cost-effective, secure logic locking," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, Apr. 2018, pp. 1–6. DOI: 10.1109/VTS.2018.8368625.

[26] B. Shakya, X. Xu, M. Tehranipoor, and D. Forte, "CAS-Lock: A Security-Corruptibility Trade-off Resilient Logic Locking Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 175–202,

2020, ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i1.175-202. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8397 (visited on 11/26/2019).

[27] R. Karmakar, S. Chatopadhyay, and R. Kapur. (Jan. 15, 2018). "Encrypt Flip-Flop: A Novel Logic Encryption Technique For Sequential Circuits." arXiv: 1801.04961 [cs], [Online]. Available: http://arxiv.org/abs/1801.04961 (visited on 04/15/2019).

[28] M. S. Rahman, A. Nahiyan, F. Rahman, S. Fazzari, K. Plaks, F. Farahmandi, D. Forte, and M. Tehranipoor, "Security Assessment of Dynamically Obfuscated Scan Chain Against Oracle-guided Attacks," *ACM Transactions on Design Automation of Electronic Systems*, vol. 26, no. 4, 29:1–29:27, Mar. 13, 2021, ISSN: 1084-4309. DOI: 10.1145/3444960. [Online]. Available: https://doi.org/10.1145/3444960 (visited on 03/18/2021).

[29] U. Guin, Z. Zhou, and A. Singh, "Robust Design-for-Security Architecture for Enabling Trust in IC Manufacturing and Test," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 818–830, May 2018, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2018.2797019.

[30] N. Limaye, E. Kalligeros, N. Karousos, I. G. Karybali, and O. Sinanoglu, "Thwarting All Logic Locking Attacks: Dishonest Oracle with Truly Random Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2020, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.3029133.

[31] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "On Designing Secure and Robust Scan Chain for Protecting Obfuscated Logic," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, Virtual Event China: ACM, Sep. 7, 2020, pp. 217–222, ISBN: 978-1-4503-7944-1. DOI: 10.1145/3386263.3407655. [Online]. Available: https://dl.acm.org/doi/10.1145/3386263.3407655 (visited on 12/16/2021).

[32] R. Karmakar, H. Kumar, and S. Chattopadhyay, "Efficient Key-gate Placement And Dynamic Scan Obfuscation Towards Robust Logic Encryption," *IEEE Transactions on Emerging Topics in Computing*, 2019, ISSN: 2376-4562. DOI: 10.1109/TETC.2019.2963094.

[33] R. Karmakar and S. Chattopadhyay, "On Securing Scan Obfuscation Strategies Against ScanSAT Attack," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, Mar. 2020, pp. 213–218. DOI: 10.1109/ISQED48828.2020.9137003.

[34] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, May 2017, pp. 95–100. DOI: 10.1109/HST.2017.7951805.

[35] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel Bypass Attack and BDD-based Tradeoff Analysis Against All Known Logic Locking Attacks," in *Cryptographic Hardware and Embedded Systems –*

CHES, W. Fischer and N. Homma, Eds., Cham: Springer International Publishing, 2017, pp. 189–210, ISBN: 978-3-319-66787-4. DOI: 10.1007/978-3-319-66787-4_10.

[36] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–122, 2019, ISSN: 2569-2925. DOI: 10.13154/tches.v2019.i1.97-122. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7335 (visited on 06/05/2019).

[37] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "KC2: Key-Condition Crunching for Fast Sequential Circuit Deobfuscation," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2019, pp. 534–539. DOI: 10.23919/DATE.2019.8715053.

[38] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal Attacks on Logic Locking and Camouflaging Techniques," *IEEE Transactions on Emerging Topics in Computing*, 2017, ISSN: 2168-6750. DOI: 10.1109/TETC.2017.2740364.

[39] L. Alrahis, S. Patnaik, F. Khalid, M. A. Hanif, H. Saleh, M. Shafique, and O. Sinanoglu, "GNNUnlock: Graph Neural Networks-based Oracle-less Unlocking Scheme for Provably Secure Logic Locking," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, Feb. 2021, pp. 780–785. DOI: 10.23919/DATE51398.2021.9474039.

[40] L. Li and A. Orailoglu, "Piercing Logic Locking Keys through Redundancy Identification," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2019, pp. 540–545. DOI: 10.23919/DATE.2019.8714955.

[41] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–14, 2021, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2021.3089555.

[42] A. Shilov. (). "Samsung Foundry: New $17 Billion Fab in the USA by Late 2023," [Online]. Available: https://www.anandtech.com/show/16483/samsung-in-the-usa-a-17-billion-usd-fab-by-late-2023 (visited on 11/18/2021).

[43] R. Kumar, "Simply Fabless!" *IEEE Solid-State Circuits Magazine*, vol. 3, no. 4, pp. 8–14, ISSN: 1943-0590. DOI: 10.1109/MSSC.2011.942448.

[44] S. Adee. (May 1, 2008). "The Hunt for the Kill Switch," IEEE Spectrum: Technology, Engineering, and Science News, [Online]. Available: https://spectrum.ieee.org/semiconductors/design/the-hunt-for-the-kill-switch (visited on 06/20/2019).

[45] (). "TechInsights Inc. - Semiconductor Analysis & IP Services," [Online]. Available: https://www.techinsights.com/ (visited on 11/14/2021).

[46] R. S. Rajarathnam, Y. Lin, Y. Jin, and D. Z. Pan, "ReGDS: A Reverse Engineering Framework from GDSII to Gate-level Netlist," in *2020*

IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Dec. 2020, pp. 154–163. DOI: 10.1109/HOST45689.2020.9300272.

[47] N. Albartus, M. Hoffmann, S. Temme, L. Azriel, and C. Paar, "DANA Universal Dataflow Analysis for Gate-Level Netlist Reverse Engineering," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 309–336, Aug. 26, 2020, ISSN: 2569-2925. DOI: 10.13154/tches.v2020.i4.309-336. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8685 (visited on 12/04/2020).

[48] L. Alrahis, A. Sengupta, J. Knechtel, S. Patnaik, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "GNN-RE: Graph Neural Networks for Reverse Engineering of Gate-Level Netlists," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021, ISSN: 1937-4151. DOI: 10.1109/TCAD.2021.3110807.

[49] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Jun. 2011, pp. 333–338.

[50] J. Baehr, A. Bernardini, G. Sigl, and U. Schlichtmann, "Machine learning and structural characteristics for reverse engineering," *Integration*, vol. 72, pp. 1–12, May 1, 2020, ISSN: 0167-9260. DOI: 10.1016/j.vlsi.2019.10.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926019303049 (visited on 03/26/2021).

[51] P. Ba, S. Dupuis, M. Palanichamy, a. G. D. Natale, and B. Rouzeyre, "Hardware Trust through Layout Filling: A Hardware Trojan Prevention Technique," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2016, pp. 254–259. DOI: 10.1109/ISVLSI.2016.22.

[52] H. Salmani, M. Tehranipoor, and J. Plusquellic, "New design strategy for improving hardware Trojan detection and reducing Trojan activation time," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, Jul. 2009, pp. 66–73. DOI: 10.1109/HST.2009.5224968.

[53] S. Dupuis, P. Ba, G. D. Natale, M. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and Hardware Trojans," in *IEEE 20th International On-Line Testing Symposium (IOLTS)*, Jul. 2014, pp. 49–54. DOI: 10.1109/IOLTS.2014.6873671.

[54] J. J. Rajendran, O. Sinanoglu, and R. Karri, "Building Trustworthy Systems Using Untrusted Components: A High-Level Synthesis Approach," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 9, pp. 2946–2959, Sep. 2016, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2016.2530092.

[55] A. Basak, S. Bhunia, T. Tkacik, and S. Ray, "Security Assurance for System-on-Chip Designs With Untrusted IPs," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1515–1528, Jul. 2017, ISSN: 1556-6013. DOI: 10.1109/TIFS.2017.2658544.

[56] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Design, Automation and Test in Europe*, Mar. 2008, pp. 1069–1074. DOI: 10.1109/DATE.2008.4484823.

[57] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Design Automation Conference*, Jun. 2012, pp. 83–89. DOI: 10.1145/2228360.2228377.

[58] R. S. Chakraborty and S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009, ISSN: 0278-0070. DOI: 10.1109/TCAD.2009.2028166.

[59] F. Koushanfar, "Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 51–63, Feb. 2012, ISSN: 1556-6013, 1556-6021. DOI: 10.1109/TIFS.2011.2163307. [Online]. Available: http://ieeexplore.ieee.org/document/5966342/ (visited on 05/03/2019).

[60] J. Dofe and Q. Yu, "Novel Dynamic State-Deflection Method for Gate-Level Design Obfuscation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 2, pp. 273–285, Feb. 2018, ISSN: 0278-0070. DOI: 10.1109/TCAD.2017.2697960.

[61] K. Juretus and I. Savidis, "Synthesis of Hidden State Transitions for Sequential Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2020, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.2994259.

[62] J. Leonhard, N. Limaye, S. Turk, A. Sayed, A. R. Díaz-Rizo, H. Aboushady, O. Sinanoglu, and H.-G. Stratigopoulos, "Digitally-Assisted Mixed-Signal Circuit Security," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03337109 (visited on 09/14/2021).

[63] M. Elshamy, A. Sayed, M.-M. Louërat, H. Aboushady, and H.-G. Stratigopoulos, "Locking by Untuning: A Lock-Less Approach for Analog and Mixed-Signal IC Security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03361417 (visited on 10/06/2021).

[64] *Overview of Design Security using Microsemi FPGAs and SoC FPGAs*, 2013. [Online]. Available: https://www.microsemi.com/document-portal/doc_view/132862-overview-of-design-security-using-microsemi-fpgas-and-soc-fpgas.

[65] J. P. Skudlarek, T. Katsioulas, and M. Chen, "A Platform Solution for Secure Supply-Chain and Chip Life-Cycle Management," *Computer*, vol. 49, no. 8, pp. 28–34, Aug. 2016, ISSN: 0018-9162. DOI: 10.1109/MC.2016.243.

[66] (). "Can The Hardware Supply Chain Remain Secure?" Semiconductor Engineering, [Online]. Available: https://semiengineering.com/

can-the-hardware-supply-chain-remain-secure/ (visited on 06/10/2019).

[67] (). "CamoGates - Reverse Engineering protection," Secure-IC, [Online]. Available: https://www.secure-ic.com/solutions/security-ips/camogates/ (visited on 08/04/2021).

[68] (). "TrustChain Security Platform," [Online]. Available: https://www.mentor.com/products/sm/trustchain-security-platform (visited on 06/11/2019).

[69] (). "EDA Forms the Basis for Designing Secure Systems," [Online]. Available: https://www.synopsys.com/implementation-and-signoff/resources/blogs/looking-past-horizon/designing-secure-systems.html (visited on 11/26/2021).

[70] S. Leef, "Automatic Implementation of Secure Silicon," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI - GLSVLSI '19*, Tysons Corner, VA, USA: ACM Press, 2019, pp. 3–3, ISBN: 978-1-4503-6252-8. DOI: 10.1145/3299874.3322803. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3299874.3322803 (visited on 10/07/2019).

[71] (). "CSAW Logic Locking Conquest," CSAW, [Online]. Available: https://www.csaw.io/logic-locking (visited on 11/16/2021).

[72] B. Tan, R. Karri, N. Limaye, A. Sengupta, O. Sinanoglu, M. M. Rahman, S. Bhunia, D. Duvalsaint, R. D., Blanton, A. Rezaei, Y. Shen, H. Zhou, L. Li, A. Orailoglu, Z. Han, A. Benedetti, L. Brignone, M. Yasin, J. Rajendran, M. Zuzak, A. Srivastava, U. Guin, C. Karfa, K. Basu, V. V. Menon, M. French, P. Song, F. Stellari, G.-J. Nam, P. Gadfort, A. Althoff, J. Tostenrude, S. Fazzari, E. Breckenfeld, and K. Plaks. (Jun. 11, 2020). "Benchmarking at the Frontier of Hardware Security: Lessons from Logic Locking." arXiv: 2006.06806 [cs], [Online]. Available: http://arxiv.org/abs/2006.06806 (visited on 06/21/2020).

[73] S. M. Plaza and I. L. Markov, "Solving the Third-Shift Problem in IC Piracy With Test-Aware Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, Jun. 2015. DOI: 10.1109/TCAD.2015.2404876.

[74] B. Colombier, L. Bossuet, and D. Hely, "Reversible Denial-of-Service by Locking Gates Insertion for IP Cores Design Protection," in *2015 IEEE Computer Society Annual Symposium on VLSI*, Montpellier, France: IEEE, Jul. 2015, pp. 210–215, ISBN: 978-1-4799-8719-1. DOI: 10.1109/ISVLSI.2015.54. [Online]. Available: http://ieeexplore.ieee.org/document/7309567/ (visited on 08/05/2019).

[75] M. T. Rahman, M. S. Rahman, H. Wang, S. Tajik, W. Khalil, F. Farahmandi, D. Forte, N. Asadizanjani, and M. Tehranipoor, "Defense-in-depth: A recipe for logic locking to prevail," *Integration*, Jan. 11, 2020, ISSN: 0167-9260. DOI: 10.1016/j.vlsi.2019.12.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926019303694 (visited on 01/20/2020).

[76] N. Rangarajan, S. Patnaik, J. Knechtel, O. Sinanoglu, and S. Rakheja, "SMART: A Secure Magnetoelectric AntifeRromagnet-Based Tamper-Proof Non-Volatile Memory," *IEEE Access*, vol. 8, pp. 76 130–76 142, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2988889.

[77] B. Colombier, L. Bossuet, V. Fischer, and D. Hély, "Key Reconciliation Protocols for Error Correction of Silicon PUF Responses," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1988–2002, Aug. 2017. DOI: 10.1109/TIFS.2017.2689726.

[78] C. Herder, M. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014, ISSN: 1558-2256. DOI: 10.1109/JPROC.2014.2320516.

[79] L.-T. L.-T. Wang, X. Wen, and K. S. Abdel-Hafez, "Design for Testability," in *VLSI Test Principles and Architectures*, Elsevier, 2006, pp. 37–103, ISBN: 978-0-12-370597-6. DOI: 10.1016/B978-012370597-6/50006-8. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780123705976500068 (visited on 02/18/2020).

[80] M. Yasin, S. M. Saeed, J. Rajendran, and O. Sinanoglu, "Activation of logic encrypted chips: Pre-test or post-test?" In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2016, pp. 139–144.

[81] M. S. Rahman, H. Li, R. Guo, F. Rahman, F. Farahmandi, and M. Tehranipoor, "LL-ATPG: Logic-Locking Aware Test Using Valet Keys in an Untrusted Environment," in *2021 IEEE International Test Conference (ITC)*, Oct. 2021, pp. 180–189. DOI: 10.1109/ITC50571.2021.00026.

[82] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Circuit Obfuscation and Oracle-guided Attacks: Who can Prevail?" In *Great Lakes Symposium on VLSI*, Banff Alberta Canada: ACM, May 10, 2017, pp. 357–362, ISBN: 978-1-4503-4972-7. DOI: 10.1145/3060403.3060494. [Online]. Available: https://dl.acm.org/doi/10.1145/3060403.3060494 (visited on 12/17/2020).

[83] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on Logic Locking: A Decade Later," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI - GLSVLSI '19*, Tysons Corner, VA, USA: ACM Press, 2019, pp. 471–476, ISBN: 978-1-4503-6252-8. DOI: 10.1145/3299874.3319495. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3299874.3319495 (visited on 05/23/2019).

[84] E. Valea, M. D. Silva, G. D. Natale, M. Flottes, and B. Rouzeyre, "A Survey on Security Threats and Countermeasures in IEEE Test Standards," *IEEE Design Test*, vol. 36, no. 3, pp. 95–116, 2019, ISSN: 2168-2356. DOI: 10.1109/MDAT.2019.2899064.

[85] M. Yasin, J. J. Rajendran, and O. Sinanoglu, "A Brief History of Logic Locking," in *Trustworthy Hardware Design: Combinational Logic Locking Techniques*, ser. Analog Circuits and Signal Processing, M. Yasin, J. J.

Rajendran, and O. Sinanoglu, Eds., Cham: Springer International Publishing, 2020, pp. 17–31, ISBN: 978-3-030-15334-2. DOI: 10.1007/978-3-030-15334-2_2. [Online]. Available: https://doi.org/10.1007/978-3-030-15334-2_2 (visited on 12/02/2020).

[86] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 66–75, Jan. 2010, ISSN: 0740-7475. DOI: 10.1109/MDT.2010.24.

[87] N. Karousos, K. Pexaras, I. G. Karybali, and E. Kalligeros, "Weighted logic locking: A new approach for IC piracy protection," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Jul. 2017, pp. 221–226. DOI: 10.1109/IOLTS.2017.8046226.

[88] R. Karmakar, H. Kumar, and S. Chattopadhyay, "On Finding Suitable Key-Gate Locations In Logic Encryption," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351235.

[89] B. Colombier, L. Bossuet, and D. Hély, "Centrality Indicators for Efficient and Scalable Logic Masking," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2017, pp. 98–103. DOI: 10.1109/ISVLSI.2017.26.

[90] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Oct. 2015, pp. 97–102. DOI: 10.1109/DFT.2015.7315143.

[91] Y. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *16th Latin-American Test Symposium (LATS)*, Mar. 2015, pp. 1–6. DOI: 10.1109/LATW.2015.7102410.

[92] L. Li and A. Orailoglu, "Shielding Logic Locking from Redundancy Attacks," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, Apr. 2019, pp. 1–6. DOI: 10.1109/VTS.2019.8758671.

[93] P. Chakraborty, J. Cruz, and S. Bhunia, "SAIL: Machine Learning Guided Structural Analysis Attack on Hardware Obfuscation," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Dec. 2018, pp. 56–61. DOI: 10.1109/AsianHOST.2018.8607163.

[94] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, Feb. 2019, ISSN: 0278-0070. DOI: 10.1109/TCAD.2018.2801220.

[95] Y. Shen and H. Zhou, "Double DIP: Re-Evaluating Security of Logic Encryption Algorithms," in *GLSVLSI 2017*, 2017. [Online]. Available: http://eprint.iacr.org/2017/290 (visited on 07/04/2019).

[96] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of Anti-SAT," in *IEEE ASP-DAC*, Jan. 2017, pp. 342–347. DOI: 10.1109/ASPDAC.2017.7858346.

[97] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, "On the Approximation Resiliency of Logic Locking and IC Camouflaging Schemes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 347–359, Feb. 2019, ISSN: 1556-6013. DOI: 10.1109/TIFS.2018.2850319.

[98] A. Rezaei and A. Mahani, "Noise-based logic locking scheme against signal probability skew analysis," *IET Computers & Digital Techniques*, vol. n/a, no. n/a, 2021, ISSN: 1751-861X. DOI: 10.1049/cdt2.12022. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cdt2.12022 (visited on 04/07/2021).

[99] J. Zhou and X. Zhang, "Generalized SAT-Attack-Resistant Logic Locking," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2581–2592, 2021, ISSN: 1556-6021. DOI: 10.1109/TIFS.2021.3059271.

[100] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly Stripping Functionality for Logic Locking: A Fault-based Perspective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.2968898.

[101] Z. Han, M. Yasin, and J. J. V. Rajendran, "Multi-Objective Strategies for Stripped-Functionality Logic Locking," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Oct. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180480.

[102] D. Sirone and P. Subramanyan, "Functional Analysis Attacks on Logic Locking," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2514–2527, 2020, ISSN: 1556-6013, 1556-6021. DOI: 10.1109/TIFS.2020.2968183.

[103] K. Juretus and I. Savidis, "Increased Output Corruption and Structural Attack Resiliency for SAT Attack Secure Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2020, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.2988629.

[104] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing Logic Encryption in Post-SAT Era by Locking Obfuscation," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2020, pp. 13–18. DOI: 10.23919/DATE48585.2020.9116500.

[105] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Jun. 2017, pp. 1–6. DOI: 10.1145/3061639.3062226.

[106] D. Duvalsaint, X. Jin, B. Niewenhuis, and R. D. Blanton, "Characterization of Locked Combinational Circuits via ATPG," in *IEEE International Test Conference (ITC)*, Nov. 2019, pp. 1–10. DOI: 10.1109/ITC44170.2019.9000130.

[107] M. E. Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2017, pp. 33–40. DOI: 10.1109/ICCAD.2017.8203757.

[108] Y. Kasarabada, S. Chen, and R. Vemuri, "On SAT-Based Attacks On Encrypted Sequential Logic Circuits," in *20th International Symposium on Quality Electronic Design (ISQED)*, Mar. 2019, pp. 204–211. DOI: 10.1109/ISQED.2019.8697421.

[109] A. Alaql, D. Forte, and S. Bhunia, "Sweep to the Secret: A Constant Propagation Attack on Logic Locking," in *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Dec. 2019, pp. 1–6. DOI: 10.1109/AsianHOST47458.2019.9006720.

[110] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "From Cryptography to Logic Locking: A Survey on The Architecture Evolution of Secure Scan Chains," *IEEE Access*, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3080257. [Online]. Available: https://ieeexplore.ieee.org/document/9431198/ (visited on 05/17/2021).

[111] X. Wang, D. Zhang, M. He, D. Su, and M. Tehranipoor, "Secure Scan and Test Using Obfuscation Throughout Supply Chain," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867–1880, Sep. 2018, ISSN: 0278-0070. DOI: 10.1109/TCAD.2017.2772817.

[112] R. Karmakar, S. Chattopadhyay, and R. Kapur, "A Scan Obfuscation Guided Design-for-Security Approach For Sequential Circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019, ISSN: 1549-7747. DOI: 10.1109/TCSII.2019.2915606.

[113] L. Alrahis, M. Yasin, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "ScanSAT: Unlocking Obfuscated Scan Chains," in *Asia and South Pacific Design Automation Conference*, (Tokyo, Japan), ser. ASPDAC '19, 2019, pp. 352–357, ISBN: 978-1-4503-6007-4. DOI: 10.1145/3287624.3287693. [Online]. Available: http://doi.acm.org/10.1145/3287624.3287693 (visited on 05/21/2019).

[114] L. Alrahis, M. Yasin, N. Limaye, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "ScanSAT: Unlocking Static and Dynamic Scan Obfuscation," *IEEE Transactions on Emerging Topics in Computing*, Sep. 10, 2019. arXiv: 1909.04428. [Online]. Available: http://arxiv.org/abs/1909.04428 (visited on 09/17/2019).

[115] N. Limaye and O. Sinanoglu, "DynUnlock: Unlocking Scan Chains Obfuscated using Dynamic Keys," in *DATE*, 2020. arXiv: 2001.06724.

[116] N. Limaye, A. Sengupta, M. Nabeel, and O. Sinanoglu, "Is Robust Design-for-Security Robust Enough? Attack on Locked Circuits with

Restricted Scan Chain Access," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[117] E. Kalligeros, N. Karousos, and I. G. Karybali, "Oracle-based Logic Locking Attacks: Protect the Oracle Not Only the Netlist," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2020, pp. 939–944. DOI: 10.23919/DATE48585.2020.9116463.

[118] Q.-L. Nguyen, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "On Preventing SAT Attack with Decoy Key-Inputs," in *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2021, pp. 114–119. DOI: 10.1109/ISVLSI51109.2021.00031.

[119] D. T. Franco, M. C. Vasconcelos, L. Naviner, and J.-F. Naviner, "Signal probability for reliability evaluation of logic circuits," *Microelectronics Reliability*, 19th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis (ESREF 2008), vol. 48, no. 8, pp. 1586–1591, Aug. 1, 2008, ISSN: 0026-2714. DOI: 10.1016/j.microrel.2008.07.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271408001947 (visited on 03/10/2022).

[120] J. Torras Flaquer, J.-M. Daveau, L. Naviner, and P. Roche, "Handling reconvergent paths using conditional probabilities in combinatorial logic netlist reliability estimation," in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*, Dec. 2010, pp. 263–267. DOI: 10.1109/ICECS.2010.5724504.

[121] S. Dupuis, G. D. Natale, M.-L. Flottes, and B. Rouzeyre, "Identification of Hardware Trojans triggering signals," in *First Workshop on Trustworthy Manufacturing and Utilization of Secure Devices*, 2013, p. 5.

[122] S. N. Pagliarini, L. A. d. B. Naviner, and J.-F. Naviner, "Selective hardening methodology for combinational logic," in *2012 13th Latin American Test Workshop (LATW)*, Quito, Ecuador: IEEE, Apr. 2012, pp. 1–6, ISBN: 978-1-4673-2355-0 978-1-4673-2354-3. DOI: 10.1109/LATW.2012.6261262. [Online]. Available: http://ieeexplore.ieee.org/document/6261262/ (visited on 03/10/2022).

[123] K. Juretus and I. Savidis, "Characterization of In-Cone Logic Locking Resiliency Against the SAT Attack," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2019, ISSN: 0278-0070. DOI: 10.1109/TCAD.2019.2925387.

[124] H. Zhou, A. Rezaei, and Y. Shen, "Resolving the Trilemma in Logic Encryption," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[125] M. Zuzak, Y. Liu, and A. Srivastava, "Trace Logic Locking: Improving the Parametric Space of Logic Locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020, ISSN: 1937-4151. DOI: 10.1109/TCAD.2020.3025135.

[126] Y. Shen, A. Rezaei, and H. Zhou, "A comparative investigation of approximate attacks on logic encryptions," in *2018 23rd Asia and South*

Pacific Design Automation Conference (ASP-DAC), Jan. 2018, pp. 271–276. DOI: 10.1109/ASPDAC.2018.8297317.

[127] K. Shamsi. (). "Netlist encryption and obfuscation suite," [Online]. Available: https://bitbucket.org/kavehshm/neos/src/master/ (visited on 11/23/2020).

[128] G. Di Natale, M.-L. Flottes, B. Rouzeyre, and P.-H. Pugliesi-Conti, "Manufacturing Testing and Security Countermeasures," in *Hardware Security and Trust: Design and Deployment of Integrated Circuits in a Threatened Environment*, N. Sklavos, R. Chaves, G. Di Natale, and F. Regazzoni, Eds., Cham: Springer International Publishing, 2017, pp. 127–148, ISBN: 978-3-319-44318-8. DOI: 10.1007/978-3-319-44318-8_7. [Online]. Available: https://doi.org/10.1007/978-3-319-44318-8_7 (visited on 12/14/2019).

[129] L. Alrahis, S. Patnaik, J. Knechtel, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "UNSAIL: Thwarting Oracle-Less Machine Learning Attacks on Logic Locking," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2508–2523, 2021, ISSN: 1556-6021. DOI: 10.1109/TIFS.2021.3057576.

[130] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design Test of Computers*, vol. 17, no. 3, pp. 44–53, Jul. 2000, ISSN: 1558-1918. DOI: 10.1109/54.867894.

[131] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury, "Secured Flipped Scan-Chain Model for Crypto-Architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2080–2084, Nov. 2007, ISSN: 1937-4151. DOI: 10.1109/TCAD.2007.906483.

[132] D. Hely, F. Bancel, M. Flottes, and B. Rouzeyre, "Test control for secure scan designs," in *European Test Symposium (ETS)*, May 2005, pp. 190–195. DOI: 10.1109/ETS.2005.36.

[133] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani, "The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, Dec. 2020, pp. 262–272. DOI: 10.1109/HOST45689.2020.9300258.

[134] Q.-L. Nguyen, E. Valea, M.-L. Flottes, S. Dupuis, and B. Rouzeyre, "A Secure Scan Controller for Protecting Logic Locking," in *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Jul. 2020, pp. 1–6. DOI: 10.1109/IOLTS50870.2020.9159730.

[135] L. Alrahis, S. Patnaik, M. A. Hanif, H. Saleh, M. Shafique, and O. Sinanoglu, "GNNUnlock+: A Systematic Methodology for Designing Graph Neural Networks-based Oracle-less Unlocking Schemes for Provably Secure Logic Locking," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2021, ISSN: 2168-6750. DOI: 10.1109/TETC.2021.3108487.

[136] C. Pilato, F. Regazzoni, R. Karri, and S. Garg, "TAO: Techniques for Algorithm-Level Obfuscation during High-Level Synthesis," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, Jun. 2018, pp. 1–6. DOI: 10.1109/DAC.2018.8465830.

[137] C. Pilato, A. B. Chowdhury, D. Sciuto, S. Garg, and R. Karri, "ASSURE: RTL Locking Against an Untrusted Foundry," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–13, 2021, ISSN: 1557-9999. DOI: 10.1109/TVLSI.2021.3074004.

[138] L. Aksoy, Q.-L. Nguyen, F. Almeida, J. Raik, M.-L. Flottes, S. Dupuis, and S. Pagliarini, "High-level Intellectual Property Obfuscation via Decoy Constants," in *2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Jun. 2021, pp. 1–7. DOI: 10.1109/IOLTS52814.2021.9486714.

[139] D. M. Luria and R. Vemuri, "Logic Encryption for Resource Constrained Designs," *IEEE Access*, vol. 9, pp. 29 312–29 345, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3059163.

[140] Y. Hu, K. Yang, S. D. Chowdhury, and P. Nuzzo, "Risk-Aware Cost-Effective Design Methodology for Integrated Circuit Locking," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, Feb. 2021, pp. 1182–1185. DOI: 10.23919/DATE51398.2021.9473956.