



Contre-mesures géométriques aux attaques exploitant les canaux cachés

Sylvain Guilley

► To cite this version:

Sylvain Guilley. Contre-mesures géométriques aux attaques exploitant les canaux cachés. domain_other. Télécom ParisTech, 2007. English. ⟨NNT : ⟩. ⟨pastel-00002562⟩

HAL Id: pastel-00002562

<https://pastel.hal.science/pastel-00002562v1>

Submitted on 25 Jun 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

GET / Télécom Paris
CNRS – LTCI (UMR 5141)



Thèse de doctorat:

Contre-mesures géométriques aux attaques exploitant les canaux cachés

Jury:

- David NACCACHE, professeur à Paris II et membre du groupe crypto de l'ENS (rapporteur)
- Jean-Jacques QUISQUATER, professeur à l'Université Catholique de Louvain-la-Neuve (rapporteur)
- Marc RENAUDIN, professeur à l'INPG (examineur)
- Pierre-Yvan LIARDET, docteur, cryptologue chez STMicroelectronics (examineur)
- Francis JUTAND, professeur, directeur scientifique du GET (président du jury)
- Yves MATHIEU, directeur d'études à l'ENST Paris (examineur)
- Renaud PACALET, ingénieur d'études à l'ENST Sophia Antipolis, responsable du laboratoire LabSoC (directeur de thèse)

Sylvain GUILLEY

January 10th, 2007

Résumé de la thèse en français

Ce travail de thèse concerne la sécurisation des circuits électroniques contre les attaques (dites « *par la bande* ») qui visent leur implémentation. Les algorithmes cryptographiques ont été traditionnellement étudiés pour résister aux attaques théoriques. Néanmoins, dès lors que ces algorithmes sont mis en œuvre sur des dispositifs concrets, de nouvelles attaques deviennent possibles. Effectivement, de l'information peut être extraite passivement (par observation) ou activement (par injection de fautes) des circuits [43]. Cette information complémentaire, communément appelée « canal caché », apporte un pouvoir supplémentaire aux attaquants. Les canaux cachés les plus populaires sont le temps de calcul, la consommation électrique, le rayonnement, la température et le bruit. Les dispositifs les plus vulnérables sont ceux qui sont alimentés de l'extérieur, comme les cartes à puce avec ou sans contacts, ou les appareils portables en général. Les radiations électromagnétiques constituent également une fuite d'information, qui permet notamment à un attaquant de réaliser des attaques à distance. Ainsi, d'une manière générale, tous les systèmes électroniques, nomades ou non, dont par ailleurs l'utilisation par le grand public ne cesse de croître, sont de potentielles cibles d'attaques malveillantes. La sécurisation du matériel est donc un enjeu sociétal. L'appropriation des systèmes électroniques embarqués (typiquement à base de TPM) ne pourra en effet perdurer que si leur niveau de sécurité apporte des garanties prouvées, inspirant par là-même aux utilisateurs finals une confiance suffisante dans la technologie.

Nous montrons tout d'abord que les attaques sur les canaux cachés (ou Side-Channel Attacks, SCA) sont des attaques structurelles, c'est-à-dire inhérentes au traitement de l'information. Il se trouve par ailleurs que les algorithmes cryptographiques sont spécialement sensibles aux attaques SCA, à cause des propriétés constitutives de certaines fonctions booléennes utilisées. Dans le cas du chiffre symétrique, nous montrons que les SCA sont inéluctables, car la fuite d'information minimale se trouve justement être égale à la clé secrète. De plus, les contraintes technologiques de mise en œuvre accentuent la force des attaques. L'étude de l'attaque montrera que le talon d'Achille principal est l'architecture RTL (*Register Transfer Level*) de l'opérateur cryptographique. Effectivement, les transferts de registres rendent possible une attaque dite en « distance de Hamming », particulièrement efficace.

Nous continuons en recherchant des moyens permettant de ne fuir pratique-

ment aucune information exploitable par un attaquant. Sur l'exemple d'un coprocesseur DES itératif, nous montrons comment exploiter concrètement les fuites d'information. Nous décrivons l'attaque par analyse différentielle de consommation (DPA) de DES en poids et en distance de Hamming sur le premier et le dernier tour de l'architecture du crypto-processeur non protégé. Des considérations subtiles, comme l'influence de la diaphonie, de la mémorisation capacitive et du moyennage temporel des traces de consommation, sont présentées. Ces effets, du second ordre, sont en effet à même d'induire de nouvelles failles de sécurité. Il s'avère donc que les portes logiques elles-mêmes possèdent des caractéristiques critiques à protéger.

Les failles de sécurité mises en évidence sur l'exemple de DES nous conduisent à émettre des recommandations pour la conception d'un opérateur robuste, basées sur l'usage d'une bibliothèque de portes et d'une stratégie de routage géométriquement équilibrées. Les portes logiques sont conçues de sorte à minimiser les violations de symétrie. Une contrainte supplémentaire a été de rendre ces portes logiques interopérables avec les portes d'une bibliothèque standard de STMicroelectronics (à savoir HCMOS9GP, du procédé 130 nanomètres). La stratégie de routage équilibré obéit aux mêmes critères. La conservation de la symétrie est traitée avec un soin tout particulier, aboutissant à la méthode générique de « *backend duplication* ». Pour autant, la compatibilité avec les outils de conception assistée par ordinateur (CAO) n'est pas négligée. La *backend duplication* s'implémente par de simples ajouts de contraintes lors du placement-routage, ce qui rend la méthode adaptable à tout flot de conception propriétaire.

La plateforme d'analyse et les logiciels d'attaques sont décrits dans les annexes du manuscrit. Les mesures de canaux cachés ont été réalisés sur des circuits coudus-main (*Application Specific Integrated Circuits*, ou ASICs) conçus dans le cadre des travaux de cette thèse. La chaîne d'acquisition s'est progressivement enrichie de fonctionnalités, à tel point qu'elle est actuellement devenue un laboratoire d'évaluation sécuritaire des systèmes embarqués. Les logiciels d'analyse de traces ont également été améliorés au cours des travaux. Les algorithmes mis en œuvre dans les attaques présentées dans ce manuscrit sont désormais plus puissants de plusieurs ordres de grandeurs.

Le reste de cette section présente les idées directrices et les enchaînements de la thèse, les résultats et les conclusions, en s'appuyant sur l'articulation logique entre les cinq chapitres.

Chapitre 1 : Cryptographie physique

Ce chapitre présente la nature des vulnérabilités des circuits électroniques, dans la situation où un attaquant est en mesure d’acquérir des mesures physiques pendant son fonctionnement. Par rapport au scénario d’une cryptanalyse conventionnelle, le contexte est donc plus riche : les attaques « logiques » sont toujours possibles, mais en plus les attaques « physiques » deviennent réalisables.

Les systèmes électroniques peuvent être modélisés par des couches d’abstraction (dites « OSI »). Les attaques, logiques comme physiques, peuvent cibler chacune de ces couches. Ceci signifie que la sécurité d’un système est égale à celle du maillon le plus vulnérable. Or il s’avère que dans le domaine des systèmes embarqués, c’est actuellement la couche la plus basse, à savoir la couche physique, qui est la plus exposée et la plus facilement attaquable en pratique. Ceci motive donc l’étude de telles attaques et la définition de conditions nécessaires pour accroître le niveau de sécurité de la couche physique.

Dans le contexte d’attaques sur la couche physique, l’attaquant dispose d’un potentiel de frappe impressionnant. Moyennant certaines hypothèses, toute variable logique peut être soit lue soit écrite [43].

Nous appelons attaques sur les canaux cachés les attaques passives, où les données internes ne peuvent qu’être consultées. Lorsque l’attaquant a le pouvoir supplémentaire de les altérer, nous parlerons d’attaques par injection de fautes. Cette catégorie d’attaque, non abordée dans le cadre de cette thèse, est également dite « active ».

Maintenant, il n’est souvent pas concevable à un prix raisonnable d’accéder arbitrairement aux variables internes des circuits cryptographiques. Les attaques physiques ne sont donc pas totalement en « boîte blanche ». Le bon niveau de modélisation est celui de la « boîte grise » :

- les canaux cachés apportent une information globale (sur plusieurs variables simultanément) et intégrée (sur plusieurs valeurs consécutives), et
- les injections de fautes touchent également plusieurs nœuds, et ce de façon non localisée dans le temps.

En résumé, les attaques physiques ne sont pas chirurgicales. Si elles l’étaient, il n’est pas difficile de comprendre qu’aucune implémentation ne leur résisterait. La barrière à franchir serait celle de la rétro-conception : il s’agit de pouvoir interpréter la masse d’information que l’on est en mesure d’accéder, pour extraire uniquement celle qui est pertinente. Néanmoins, des techniques de fouille de données (*data mining*) éprouvées permettent aisément de répondre à ce problème de gestion de la complexité.

Par ailleurs, les données seront souvent bruitées. La mesure physique, surtout de quantités nanoscopiques, n’est pas parfaite. De même, l’injection de fautes ne peut pas être contrôlée avec une précision arbitraire. Ce caractère aléatoire conduit à la définition d’attaques statistiques. Nous investiguerons surtout l’analyse différentielle de consommation, qui consiste en une corrélation statistique

d'un jeu de données avec un modèle de fuite. D'autres attaques statistiques ne nécessitent aucune connaissance préalable du circuit attaqué. Il s'agit par exemple de l'IPA (*Inferential Power Analysis* [32]) ou des attaques par patron (aussi appelées *template attacks* [25]). Néanmoins, celles-ci nécessitent une phase d'entraînement sur un circuit clone pour acquérir de la connaissance sur le fonctionnement de la victime. Nous n'aborderons pas ces attaques – néanmoins prometteuses – dans ce manuscrit.

À l'inverse, nous supposons dans ce travail que non seulement l'algorithme utilisé, mais également le dessin des masques de son implémentation, sont connues de l'attaquant. Il s'agit donc du scénario d'attaque le plus flexible. Dans le même ordre d'idée, nous autoriserons un attaquant à faire un nombre d'appels non borné aux opérations cryptographiques. Cela signifie que l'on suppose que toutes les contre-mesures « haut-niveau » ont été désactivées ou contournées. Bref, le niveau de sécurité envisagé est celui d'une implémentation nue, qui doit se défendre simplement au niveau où on l'attaque, à savoir sur la couche physique.

Pour évaluer des contre-mesures, nous avons réalisé des prototypes de circuits cousus main (ASIC – *Application Specific Integrated Circuits*). Dans la même optique d'évaluation sécuritaire que celle mentionnée plus haut, ces circuits donnent à l'attaquant un signal de synchronisation avec les opérations cryptographiques et permettent un accès aisé aux canaux cachés (par exemple la consommation électrique instantanée ou le rayonnement électromagnétique).

Dans la conception des contre-mesures, la priorité a été résolument mise sur la sécurité, en évitant tout compromis sur les performances. La motivation de cette approche radicale est simple : notre démarche est de prouver qu'il existe une façon de rendre impossible les attaques sur les canaux cachés. Si, même en investissant le prix maximum, en terme de temps de calcul, de surface d'implémentation et d'énergie consommée, notre solution reste faillible face aux attaques passives, alors aucun compromis ne sera résistant. À l'inverse, si cette solution est fructueuse, il pourra être envisagé d'optimiser les performances, en réduisant de façon concomitante le niveau de sécurité, que l'on suppose donc surévalué. Une telle étude ouvrira la porte à de nombreuses solutions technologiques, intéressantes à deux points de vue :

1. Il s'agira de la première fois qu'un nouveau critère, à savoir la « sécurité », pourra être intégré au triangle canonique — vitesse, surface, consommation — de l'adéquation algorithme-architecture ; En d'autres termes, des options de sécurité vont pouvoir émerger dans les outils logiciels CAO de conception automatique de circuits.
2. La sécurité pourra être mesurée en unité monétaire, et également pour la première fois, il sera concevable de dégrader le niveau de sécurité au profit des performances, alors que jusqu'à présent, dans la conception des cryptoglyphes, la sécurité est considérée comme un attribut non négociable.

Les analyses de canaux cachés fournies dans cette thèse sont rassemblés en annexe, afin ne pas surcharger le corps du document. Contrairement à la grande ma-

porité des publications scientifiques du domaine, nous avons pris soin de préciser les unités de grandeurs physiques et de resynchroniser les traces avec le déroulement de l’algorithme cryptanalysé. Il s’agit de la première étude en consommation précise au cycle près de matériel électronique. La résolution temporelle est de loin supérieure à ce qui est *stricto sensu* nécessaire pour réussir une attaque : les acquisitions sont réalisées au rythme de 20 milliards de point par seconde sur des circuits tournant à 32 ou 66 MHz. Ainsi, plusieurs centaines d’échantillons sont disponibles par période d’horloge.

Enfin, il convient de comparer la force des attaques physiques et logiques. Si l’on ne considère que le temps de traitement des données, voilà deux chiffres permettant de comparer la puissance relative des deux types d’attaques sur l’algorithme de chiffrement symétrique DES [71] :

- un ordinateur de bureau typique est capable d’exécuter $5\,369\,726 \sim 2^{22}$ chiffrements DES par seconde (performance obtenue avec la commande “`openssl speed des`”),
- un oscilloscope met une seconde pour l’acquisition d’une trace de consommation (cas très défavorable avec du matériel bas de gamme).

Cela signifie qu’une attaque logique (la recherche exhaustive des 56 bits de la clé) est aussi puissante qu’une attaque physique (une analyse différentielle de consommation) si l’on réussit celle-ci en $2^{56-22} = 2^{34}$ traces, soit environ 17 milliards de traces. Or, sur les implémentations non protégées, les attaques physiques réussissent souvent après analyse de moins de mille (disons 2^{10}) traces. Les résultats du chapitre 3 démontrent cette assertion. La conclusion est donc que les attaques physiques sont, dans l’état de l’art actuel, de plusieurs ordres de grandeurs plus puissantes que les cryptanalyses logiques. Le danger qu’elles représentent est donc bel et bien réel, ce qui motivera au chapitre 4 la recherche de contre-mesures efficaces.

Chapitre 2 : Immanence des canaux cachés

Ce chapitre montre que, quelque soit l'algorithme de chiffrement utilisé, une analyse par les canaux cachés permet d'extraire l'intégralité de la clé. Tout d'abord, un modèle de dissipation logique est présenté. Lorsque des contre-mesures appropriées ne sont pas prises, le modèle démontre que l'information minimale qui est fuie se trouve être la clé. Ensuite, un modèle de l'attaque est présenté. Il montre qu'une attaque sur les canaux cachés est d'autant plus puissante que l'algorithme attaqué est robuste du point de vue cryptanalytique. La validité du modèle s'étend aussi bien aux algorithmes à clé secrète de type Feistel (comme DES [71]) qu'aux réseaux de permutation-substitution (comme AES [72]). Enfin, ce chapitre se termine sur des remarques importantes au sujet de l'implémentation pratique de l'analyse différentielle de consommation.

Nous qualifions une attaque de « structurelle » quand elle s'applique à une classe générique d'algorithmes. Les attaques sur les canaux cachés sont structurelles, car elles concernent tous les algorithmes qui dissipent une quantité physique corrélée à un secret. Nous montrons comment un algorithme de chiffrement symétrique tel que DES est obligé de fuir de l'information. Dans la figure 1, il apparaît que :

- (a) soit l'algorithme dissipe *logiquement* la clé secrète k ,
- (b) soit il est spécialisé pour une clé k_0 , et s'expose par là même à une rétro-conception ou à une divulgation publique de son implémentation.

Ainsi, le cas (b) est naturellement exclus par le principe de Kerkhoff. Le cas (a) montre qu'il est nécessaire de détruire la clé k avant de diffuser la sortie (c, k) de la boîte chiffiante. Cette destruction d'une entropie de 56 bits s'accompagne physiquement d'une dissipation corrélée à la clé, car celle-ci est nécessairement stockée dans une ressource matérielle (registre, mémoire, *etc.*) qui devra être mise à zéro ou masquée par toute autre grandeur indépendante de la clé.

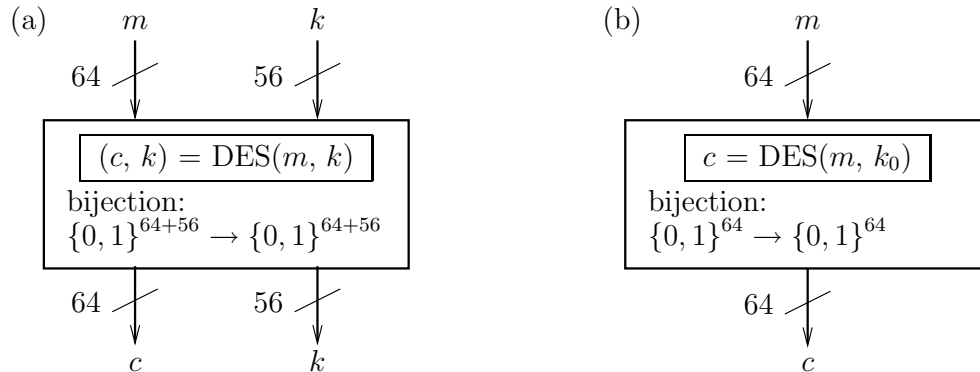


FIG. 1 – Conception abstraite d'un chiffrement symétrique (ici, le crypto-système DES [71]), (a) générique dans la clé k ou (b) personnalisée pour une clé donnée de 56 bits k_0 .

Il existe certes des méthodes théoriques permettant de réaliser une opération sans dissiper d'énergie sur la durée totale du calcul. Deux façons de procéder s'appuient sur des styles de logiques proposés par Toffoli et Fredkin :

1. La logique réversible [107] consiste à décomposer les bijections booléennes vectorielles sur des primitives *a priori* non dissipatives.
2. La logique conservative [35] consiste à calculer à poids de Hamming constant. Un modèle de calcul, dit « à boules de billard », permet effectivement d'imaginer un mode de calcul sans dissipation.

Néanmoins, aucune implémentation pratique de ces logiques n'a encore vu le jour, ce qui motive une recherche de solutions avec les procédés technologiques actuels, comme la logique CMOS [65] par exemple.

Un modèle de l'attaque DPA est présenté. Il s'appuie sur la modélisation simple de la consommation : toutes les portes consomment le même quantum d'énergie quand elles commutent. Il s'agit donc d'un modèle basé sur l'étude de l'activité logique. Lorsque ce modèle est appliqué à l'attaque DPA, qui cible soit le premier, soit le dernier tour, il se trouve que les opérations linéaires n'interviennent pas. Seule les opérations non-linéaires, qui se réduisent dans la plupart des cas à l'utilisation d'une table de substitution (dite « sbox », et notée F ou S , opérant de $\{0, 1\}^p$ dans $\{0, 1\}^q$) impacte le modèle d'attaque. Les résultats [89] sont que :

- la corrélation maximale est obtenue pour l'hypothèse de clé correcte. Cette observation provient de l'application du théorème de Cauchy-Schwarz sur la fonction pseudo-booléenne d'auto-corrélation.
- Les corrélations sont, en moyenne, nulles,
- ce qui prouve qu'il existe des corrélations pour les hypothèses de clé erronées (appelée « pics fantômes » – ou aussi « *ghost peaks* » dans la littérature scientifique anglaise).
- La qualité de l'attaque peut donc se mesurer par un rapport signal-à-bruit (SNR), qui ne dépend que de la structure de l'algorithme, à savoir aux propriétés des boîtes de substitution utilisées.
- De plus, le rapport signal-à-bruit ne dépend pas de la clé, ce qui démontre qu'il n'y a pas de clé faible en attaque DPA.

Le rapport entre le SNR de l'attaque DPA et les sbox est caractérisé par les deux relations suivantes, qui définissent des bornes inférieures :

$$\begin{aligned} \text{SNR(DPA)}(F) &\geq \frac{2^{\frac{3p}{2}-2}}{q \Lambda_S^2} = \mathcal{O}\left(\frac{1}{\Lambda_S^2}\right), \\ \text{SNR(DPA)}(F) &\geq \frac{2^p}{\Delta_S} = \mathcal{O}\left(\frac{1}{\Delta_S}\right). \end{aligned}$$

Les deux quantités Λ_S et Δ_S , définies dans [34], sont respectivement les caractéristiques linéaires et différentielles des sboxes. Elles sont d'autant plus petites que la sbox est solide face aux cryptanalyses linéaires ou différentielles. Ainsi, mieux un système de chiffrement est protégé contre les attaques logique, plus facilement il est attaquable via les canaux cachés.

(a) Inter-pénétration de cônes logiques (b) Cône logique interne vers le nœud n_0 FIG. 2 – Illustration de la factorisation des cônes de logique sur une fonction combinatoire de 4 bits vers 4 bits ($a[0..3] \rightarrow a'[0..3]$).

Une contre-mesure possible contre la DPA serait d'agrandir la taille des boîtes de substitution ou de faire en sorte que de nombreux (idéalement tous) bits de clé interviennent dans le calcul de confusion (rôle dévolu aux sbox) dès le premier tour. Ces contre-mesures souffrent néanmoins d'un problème d'intégrabilité. Toutes deux accroissent la taille de l'implémentation de façon exponentielle, ce qui est rédhibitoire. Ainsi, c'est la nécessité de pouvoir scinder un algorithme en sous-parties de taille raisonnable (par exemple 8 bits) qui rend possible des attaques exhaustives partielles, qui une fois réunies, compromettent l'intégralité du système. Les attaques sur les canaux cachés fonctionnent effectivement sur la base d'une stratégie dite « *diviser pour régner* » ou en terminologie anglo-saxonne « *divide-and-conquer* ».

Enfin, il n'a pas encore été prouvé qu'une attaque à *l'intérieur* d'un cône de logique ne soit pas réalisable. Dans ce cas de figure, un attaquant peut sonder par les techniques statistiques de la DPA l'activité d'un nœud arbitraire, comme n_0 au cœur même de la mer de portes constituant la sbox (voir la figure 2). Dans ce cas, l'attaque se ramène à un test exhaustif d'hypothèses, mais sur un nombre réduit de bits de clé de tour.

En conclusion, si la constitution d'algorithmes intrinsèquement résistants aux attaques sur les canaux cachés est une problématique porteuse, les solutions les plus naïves ne sont certainement pas pérennes. Ceci laisse à penser que les contre-mesures « bas niveau » ont un potentiel moins incertain.

Dans le chapitre 3, l'attaque DPA est étudiée dans le cas de DES (sbox $p = 6 \rightarrow q = 4$). Les vulnérabilités constatées permettent de définir un cahier des charges sécuritaire, qui sera appliqué dans le chapitre 4 pour concevoir et implémenter un style de logique résistant de façon optimale aux fuites d'information via la consommation électrique.

Chapitre 3 : Attaque DPA sur l'algorithme DES

L'objectif de ce chapitre est de montrer quels sont les facteurs qui rendent possibles une attaque de type DPA. Contrairement au chapitre précédent, de nature théorique, ce chapitre se veut expérimental. Les vulnérabilités ne sont pas toutes capturées par un modèle, aussi sophistiqué soit-il. La confrontation de la théorie avec la pratique conduit à des enseignements sur la nature des failles exploitables par un attaquant.

Parmi les circuits qui sont potentiellement les cibles d'attaques physiques, on peut citer les ASIC, les FPGA, les micro-processeurs, ou encore les générateurs d'aléa vrai. Nous concentrons notre étude sur les accélérateurs matériels réalisés sur-mesure (ASIC). Notons que dans tous les cas de figure, il est souvent inutile de sécuriser toute l'application, sans discernement. À l'inverse, il s'agit d'identifier l'élément qui pourra être le germe de la sécurité, qui se propagera grâce à des protocoles idoines à l'ensemble du système. Ainsi, à titre d'exemple, nous envisageons l'étude de cas d'un crypto-processeur DES.

DES est actuellement l'algorithme de chiffrement symétrique le plus étudié par la communauté scientifique et le plus répandu dans les systèmes embarqués (il est notamment utilisé dans les masques B0' des cartes bancaires, *etc*). L'algorithme utilise huit boîtes de substitution S1–S8 différentes, ce qui permet de tester la force des attaques en fonction des caractéristiques de chacune d'entre elles. Néanmoins, le choix de DES pour l'étude de vulnérabilité aux attaques sur les canaux cachés n'est pas « verrouillant ». Effectivement, les conclusions de ce chapitre peuvent se transposer à tout système cryptographique utilisant des boîtes de substitution. Seul le cas de RC5 est *a priori* non adressé par nos résultats.

L'architecture du module de chiffrement DES est décrite de façon détaillée dans l'article de revue [101]. Il est nécessaire de bien connaître à la fois les ressources matérielles (niveau RTL) et l'ordonnancement des opérations, car celui-ci conditionne les transferts de registres. Le schéma du co-processeur étudié est résumé dans la figure 3.

Il est notoirement connu que DES a deux problèmes de sécurité [63] (hormis la trop faible longueur des clés) :

1. Il existe des clés faibles, pour lesquelles le chiffrement est involutif. Avec de telles clés, il existe 2^{32} points fixes.
2. De plus, il existe des couples de clés semi-faibles, telles que le chiffrement avec l'une corresponde au déchiffrement avec l'autre. En utilisant des telles clés, il existe 2^{32} anti-points fixes.

Ces propriétés permettent à un attaquant de localiser dans le temps le début et la fin d'un chiffrement. Par exemple, la figure 4 permet de mettre en évidence :

- le chargement de la clé (cycles notés 0–8),
- le chiffrement lui-même (cycles notés 16–32).

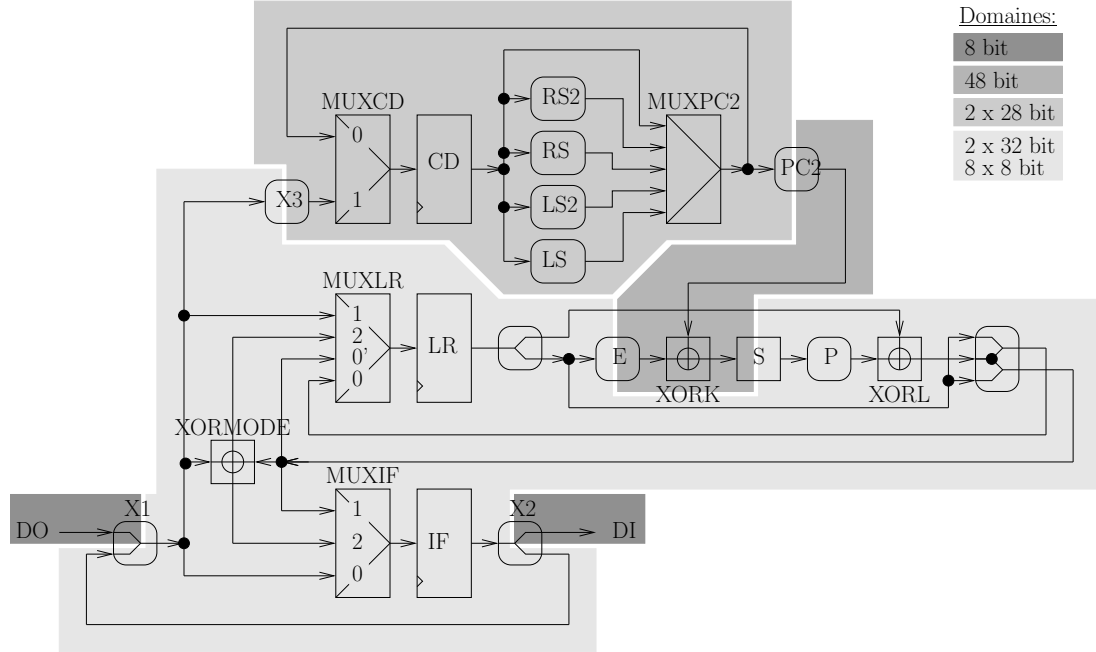


FIG. 3 – Version « SecMat V1 » du chemin de données de l'accélérateur de chiffrement DES, avec indication de la largeur des bus (en teintes de gris.)

Quand les clés ne peuvent pas être choisies par l'attaquant, l'existence de clés particulières n'est pas un problème. Nous réalisons alors avec des clés « banalisées » les attaques DPA en poids et en distance de Hamming. Les deux attaques réussissent. Néanmoins, il s'agit de noter les observations suivantes :

1. L'attaque en poids de Hamming réussit car il existe une signature dépendant de la clé au tour suivant celui que l'on pense initialement attaquer. Nous montrons que c'est l'existence d'un *glitch* qui cause la signature sans équivoque du poids de Hamming de la sortie des huit sboxes. Cette constatation va inciter la mise en place de contre-mesures où les portes élémentaires re-synchronisent entre elles leurs entrées, afin d'interdire la propagation de transitions non fonctionnelles.
2. L'attaque en distance de Hamming entre parfaitement dans le cadre du modèle présenté dans le chapitre 2, et réussit donc sans surprise particulière. Nonobstant, alors qu'un unique pic de corrélation est attendu, on en observe en réalité trois. L'explication de la complexité *a priori* inattendue de ce phénomène provient de la structure dite de Feistel de l'algorithme DES. À chaque itération, la moitié droite du message sert à la fois à calculer un masque pour la partie gauche, et à être conservée telle quelle pour, afin que le système reste déchiffirable. Le « schéma de Feistel » est représenté dans la figure 3.19 à la page 72. Ainsi, l'existence de cette *fourche* sur la moitié droite du message provoque, en plus du pic principal, une duplication de la

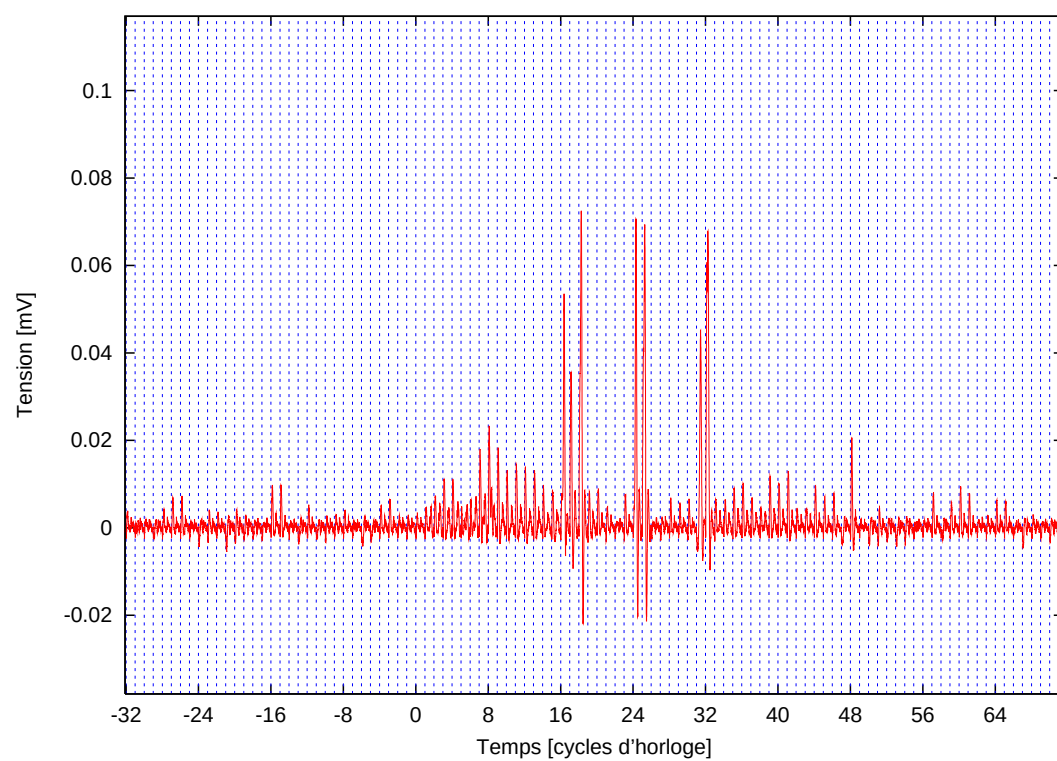


FIG. 4 – Analyse SPA obtenue par différence de seulement deux courbes de consommation de DES, utilisant une clé faible et une autre semi-faible.

fuite, ayant lieu pour moitié dans le tour précédent et pour moitié dans le tour suivant. Ceci explique complètement que trois pics soient observés.

D'autres types d'analyses, notamment à clé connue, sont réalisées. Elles permettent de tirer les conclusions suivantes quant aux caractéristiques sensibles d'une implémentation matérielle :

- Les attaques en poids de Hamming peuvent réussir (même sans état initial fixé) par la simple propagation d'un *glitch*. Il s'agit donc de supprimer les *glitches* et également d'équilibrer les portes logiques dans leurs entrées.
- Certaines corrélations peuvent se manifester alors qu'elles ne sont pas attendues. Il est ainsi nécessaire de bien étudier les transferts de données à l'intérieure d'une zone de confiance et également aux interfaces avec les zones non sécurisées.
- Il existe des corrélations au-delà de deux périodes d'horloge. Cela va inciter à supprimer les états « haute impédance » (aussi dits « Z ») et à utiliser des protocoles de calcul reposant sur une alternance pré-charge / évaluation.
- La diaphonie peut être mise en évidence. En conséquence, les équipotentielles sensibles doivent être protégées par du blindage avec un nœud global.
- Les attaques continuent à réussir même sur des traces moyennées dans les temps. Les contre-mesures consistant à introduire une gigue temporelle sont donc inopérantes. Il faudra donc concentrer les efforts vers une logique *déterministe* (par opposition à *aléatoire*).

Chapitre 4 : Contre-mesures au niveau de la conception physique

Les vulnérabilités identifiées dans le chapitre précédent sont autant de recommandations sécuritaires pour la définition et la mise en place de contre-mesures pertinentes. Nous spécifions et implémentons dans ce chapitre une réalisation possible de calcul qui pallie à tous les écueils identifiés.

En technologie VLSI, le calcul numérique se décompose en deux étapes :

1. Le grain élémentaire de calcul, réalisant des opérations booléennes partielles.
2. L'interconnexion des résultats booléens indépendants pour réaliser la fonctionnalité globale.

Ces deux étapes sont étudiées successivement. La première porte sur le dessin des masques d'une famille de portes logiques. Quant à la seconde, elle concerne l'utilisation éclairée des outils de placement-routage. L'objectif de la deuxième étape est d'éviter de ruiner les efforts réalisés localement lors de leur assemblage final sur silicium.

La partie logique s'appuie sur une représentation redondante des données, dite « *dual-rail* ». Chaque nœud logique A est représenté par une paire de fils (a_0, a_1) , dont l'évolution dans le temps est décrite comme ci-après :

Phase	NULL	→	VALID(0,1)	→	NULL	→	VALID(0,1)	→	...
(a_0, a_1)	(0,0)	→	(0,1)	→	(0,0)	→	(1,0)	→	...

Dans ces chronogrammes, la valeur de pré-charge a été fixée arbitrairement à $(0,0)$. Les deux valeurs valides sont $A = 0 \Leftrightarrow (a_0, a_1) = (1,0)$ et $A = 1 \Leftrightarrow (a_0, a_1) = (0,1)$. Cependant, d'autres représentations peuvent être utilisées avantageusement. Par exemple, la logique dite « à espaceurs entrelacés » [96] utilise de façon interchangeable $(0,0)$ ou $(1,1)$.

Les portes utilisées, respectant le protocole *dual-rail* avec retour à l'état NULL, défini par $\text{NULL} \doteq (0,0)$, doivent de plus être équilibrées vis-à-vis des chemins d'exécution. Nous proposons la structure illustrée dans la figure 5 pour le cas particulier de la porte universelle NON-ET. La porte réalise successivement les étapes suivantes :

1. Synchronisation des entrées, par couples, grâce à des portes de « rendez-vous », aussi connues sous le nom de portes de Muller ou C-éléments [93].
2. Calcul à proprement parler : aiguillage vers la sortie *vraie* ou *fausse* en fonction du min-terme qui a été décodé.

Afin d'accroître le niveau de robustesse de cette porte, trois niveaux de sécurité supplémentaires sont mis en œuvre :

1. Les portes de rendez-vous sont toutes identiques, et sans état haute-impédance suspicieux. Il est à noter que seules les équipotentielles internes N_a et N_b doivent être protégées, car P_a et P_b ne fuient pas d'information (*confer* figure 5).

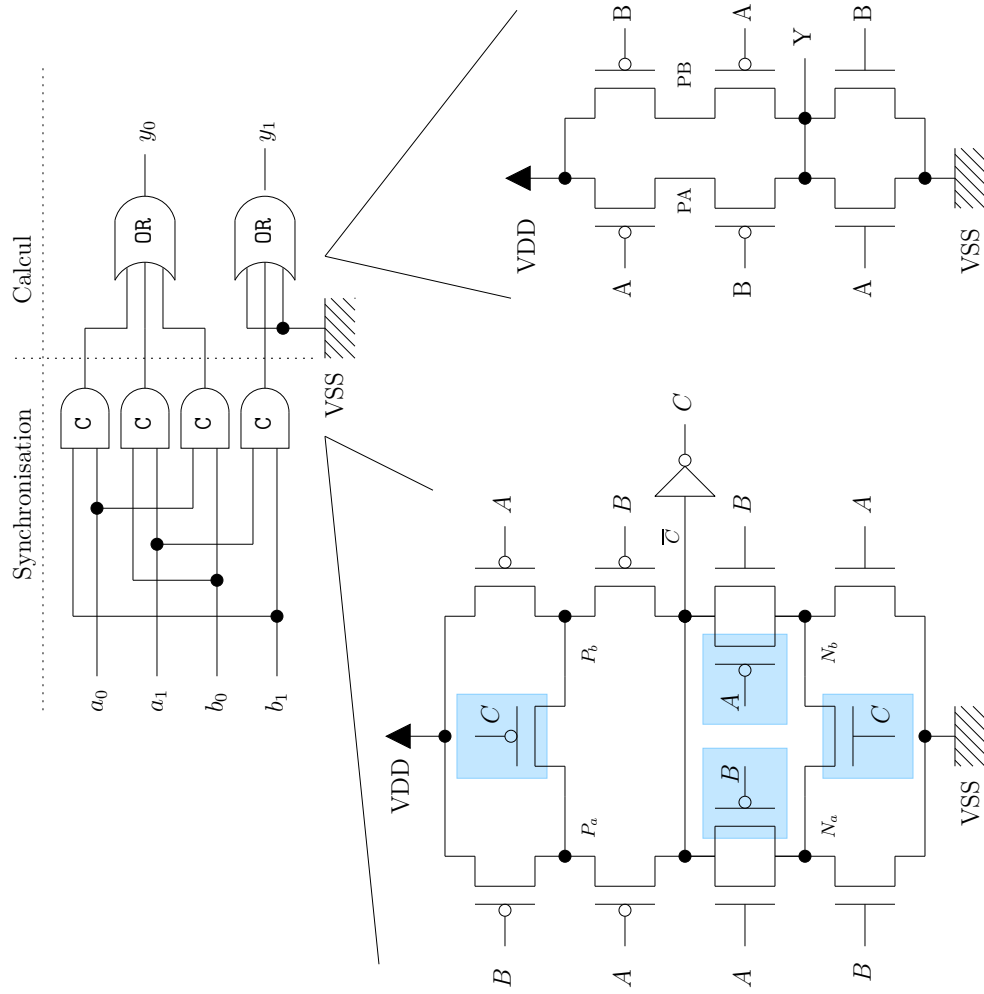


FIG. 5 – Schéma en transistor de la porte NON-ET sécurisée [42] de la bibliothèque SecLib.

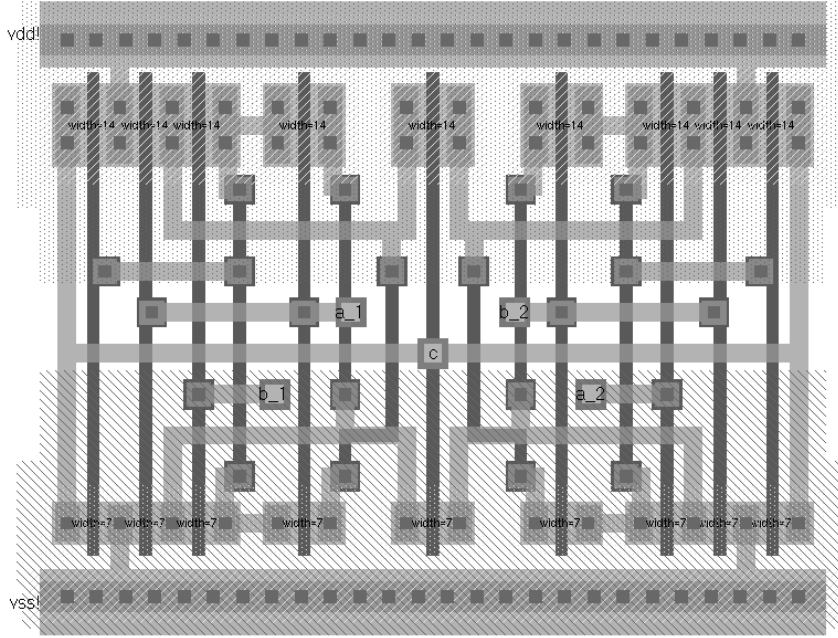


FIG. 6 – Dessin des masques de la porte de rendez-vous symétrisée et équilibrée, sans aucun état haute-impédance.

2. Les portes de concentration du calcul sont de type OR. Leur structure est rendue redondante, pour éviter de particulariser l'une des deux entrées.
3. Au niveau global, alors que, *stricto sensu*, seules quatre portes OR sont nécessaires pour véhiculer les quatres valeur de la table de vérité vers les deux sorties, six (trois et trois) sont utilisées en pratique. De cette façon, les deux sorties sont chargées par le même nombre de portes, et le temps de calcul devient constant.

Un enjeu important est de conserver la symétrie en passant du *schéma* au *dessin des masques*. Cette étape est réalisable, moyennant certaines concessions bien identifiées. La structure, dite « symétrique », de la porte de rendez-vous, se prête particulièrement bien à une projection technologique avec respect des invariants topologiques, comme illustré dans la figure. 6.

Le résultat final sur la fonction NON-ET est une porte duale, dessinée dans la figure 7.

Le niveau d'indiscernabilité atteint par cette porte est validé par simulation électrique exhaustive de tous les cas de figure (arrivées concomitantes ou différées, dans n'importe quel ordre autorisé par le protocole). La simulation confirme bien que la signature en consommation de la porte est indépendante des données manipulées.

Le dessin des masques de toutes les portes sécurisées à deux entrées est identique, *modulo* la position de certains vias. Ceci permet de concevoir rapidement

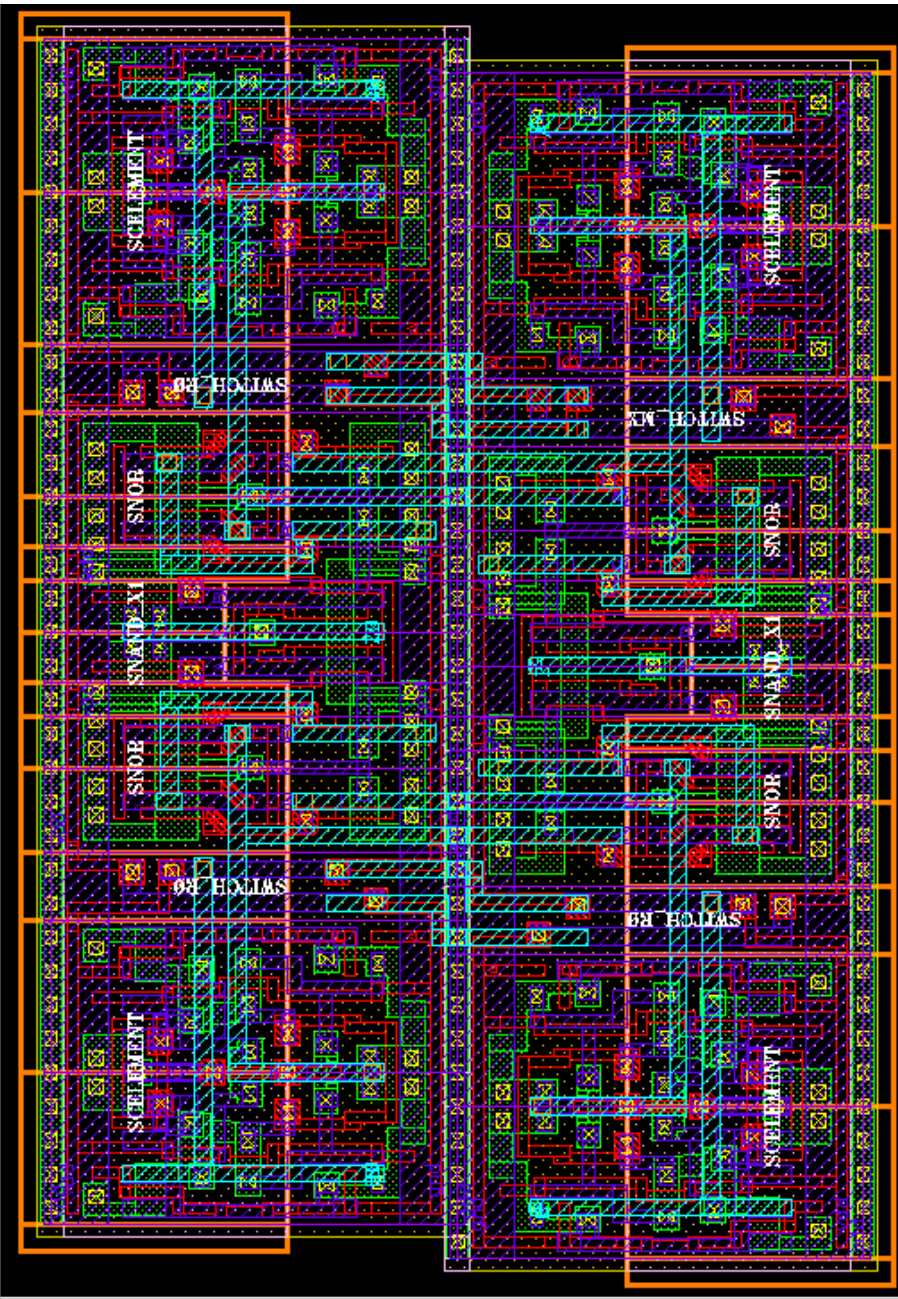


FIG. 7 – Exemple du dessin des masques de la porte **NON-ET** de la bibliothèque SecLib, vue depuis l'éditeur Virtuoso de Cadence[®].

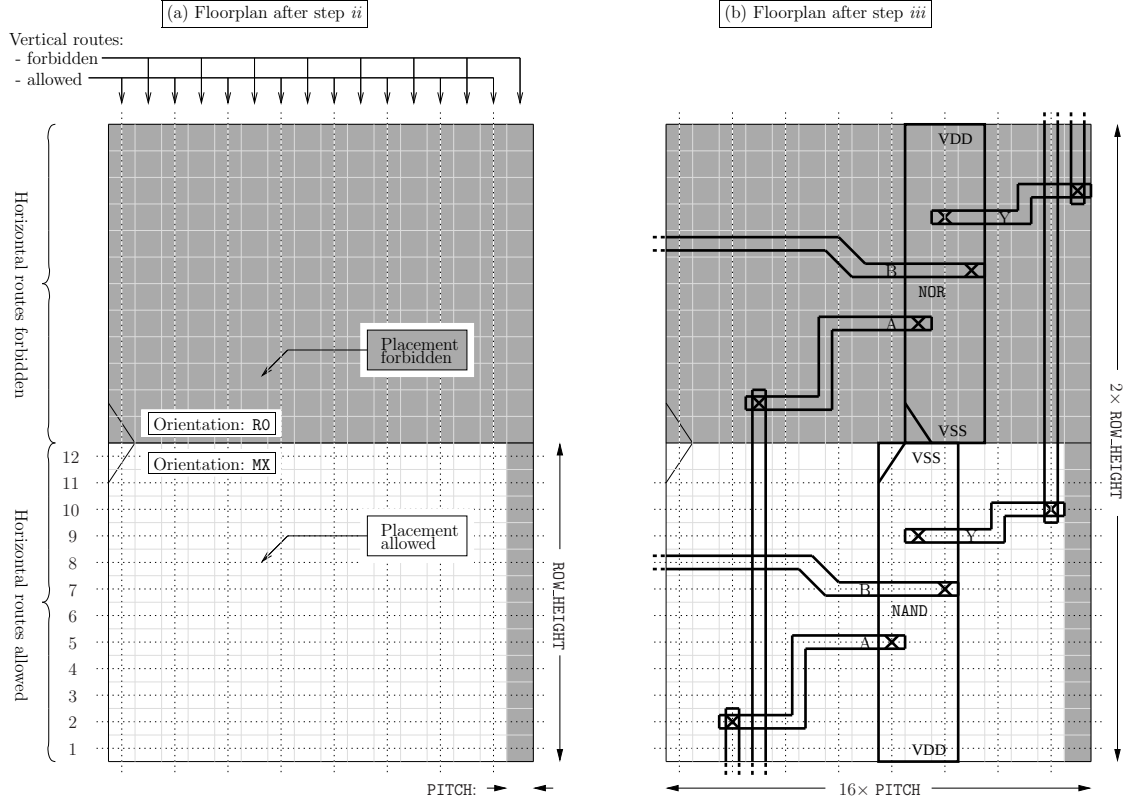


FIG. 8 – Plan de masse en *dual-rail* pour la porte NON-ET sécurisée [42].

une bibliothèque, surnommée « SecLib », comportant les 10 portes logiques à deux entrées non dégénérées (parmi les $2^{(2^n)}$ possibles, avec $n = 2$).

Les deux principales étapes de la conception physique, à savoir le placement et le routage, se doivent de préserver les invariants de sécurité mis en place dans les portes logiques de SecLib. Grâce à un placement différentiel (chaque instance voit son homologue dual dans une symétrie axiale) et à un routage également différentiel (chaque fil est le translaté de son dual), les contraintes géométriques de sécurité sont étendues du niveau *porte* au niveau *netlist* (composition de portes).

La figure 8 illustre l'opération sur une *netlist* simplifiée, réalisant comme seul calcul la fonction NON-ET.

L'opération illustrée dans la figure 8 est en pratique réalisée par l'adjonction de trois étapes à un flot de conception usuel. Ces étapes sont :

1. Le dimensionnement préalable du plan de masse à des distance paires en terme d'espacement de routage (ou « *pitch* ») et de sites de placement.
2. Avant le placement et le routage, l'instanciation d'obstructions de placement sur une rangée de placement sur deux, et la mise en place de contraintes de routage :
 - les canaux horizontaux sont interdits au-dessus des zones de placement interdites et

- seul un canal vertical sur deux est accessible.
- 3. À la fin du flot de placement-routage contraint, l'ensemble des éléments (placement, routage, signaux globaux, métal non-fonctionnel de remplissage ou *dummies*, etc.) sont dupliqués par un script *ad hoc*, traitant par exemple la description DEF [56] du circuit.

La méthode proposée dans ce chapitre a été mise en œuvre sur le chemin de donnée de l'opérateur DES décrit dans le chapitre 3. Les résultats sont résumés dans la figure 9.

Le premier histogramme (à gauche de la figure 9) représente la répartition des rapports $C(\text{true}) / C(\text{false})$ des équipotentielles de chaque paire différentielle. La répartition est toujours centrée autour de 1. Mais la dispersion diminue notamment lorsque le placement-routage est contraint :

- Sans contrainte, la dispersion peut monter jusqu'à des rapports 1:4.
- En contraignant simplement le placement (les instances duales sont côte à côte), les écarts sont fortement réduits. Cette stratégie est porteuse en technologie FPGA, où il est difficile (voire impossible avec certains IDE) de contraindre le routage.
- Avec la méthode de duplication du plan de masse [102], l'équilibrage gagne encore un ordre de grandeur.

Le test $C(\text{true}) / C(\text{false})$ est particulièrement significatif, car nous avons évalué que dans la technologie utilisée et avec le crypto-système DES, la contribution majoritaire des capacités d'interconnexion provient du routage, et non plus des capacités d'entrée des portes.

Le second histogramme (à droite de la figure 9) présente la dispersion des résistances des fils. Dans le cas de la duplication du plan de masse [102], la répartition est un pic de Dirac parfait, car par construction, les fils de chaque paire ont exactement la même longueur. Et comme il n'existe pas de couplage entre résistances (à l'inverse des capacités, pour lesquelles la diaphonie est un phénomène non négligeable), la distribution n'a pas de variance.

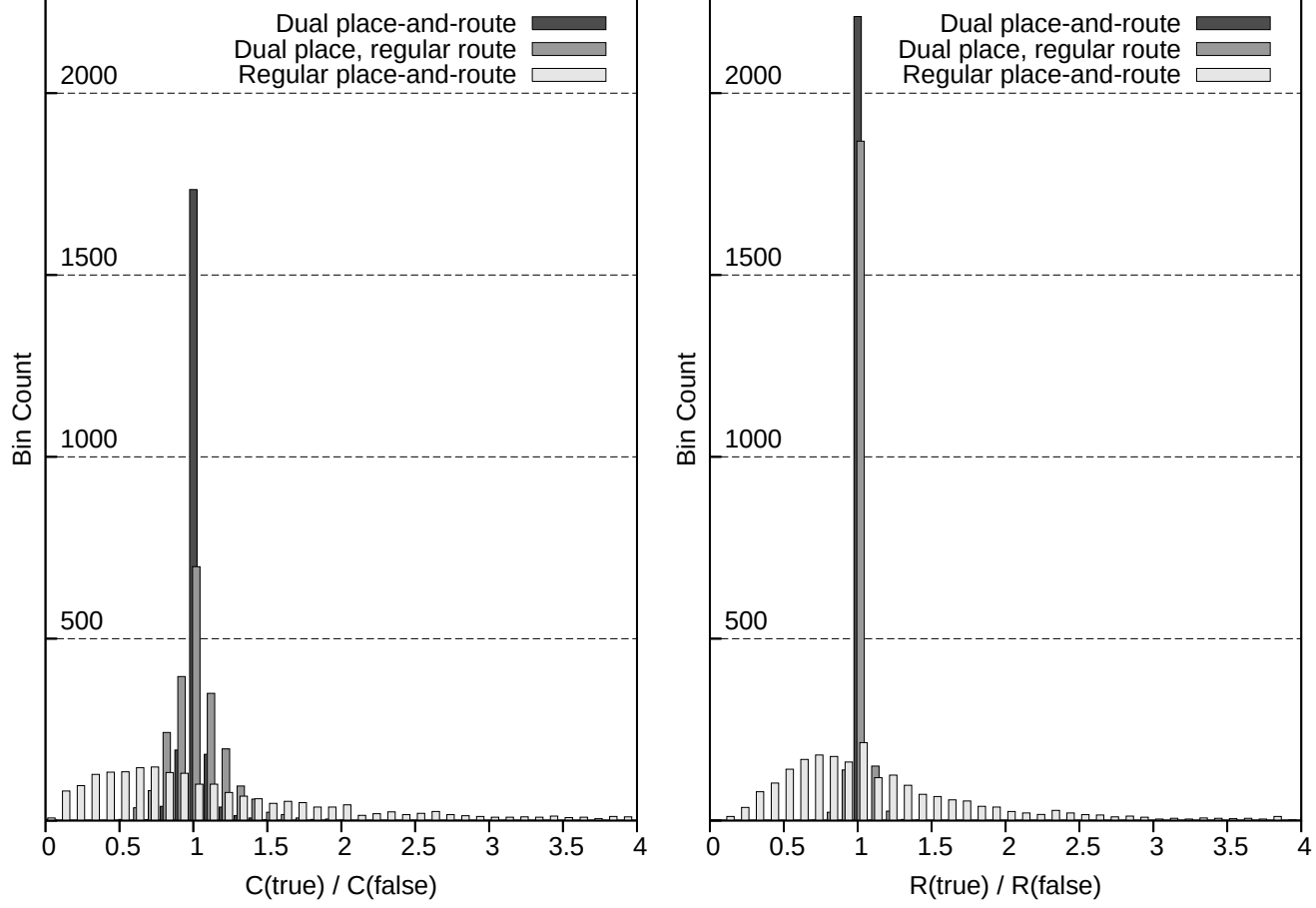


FIG. 9 – Rapport des couples de capacité et de résistance des lignes *dual-rail* du chemin de donnée de l'opérateur DES sécurisé.

Chapitre 5 : Conclusions

Depuis notre entrée de plein pied dans la « société de l'information », la grande majorité des échanges de données est devenue numérique. En conséquence, les appareils électroniques qui manipulent de l'information sont devenus omniprésents. Ils ont envahi aussi bien la sphère professionnelle que privée. Dans ce contexte, dit de « calcul pervasif », les problèmes de sécurité s'avèrent être cruciaux. Jusqu'à très récemment, les attaques usuelles ciblaient les *bugs* logiciels ou les faiblesses des protocoles des réseaux. Maintenant que les terminaux (mobiles, assistants personnels, ordinateurs de bureau, *etc.*) se rapprochent de l'utilisateur final, l'implémentation elle-même est devenue la cible d'attaques. Ceci s'oppose à la conception traditionnelle de la sécurité, où les organes sensibles (serveurs, routeurs, *etc.*) pouvaient être considérés comme des forteresses. De plus, la sécurité devient de moins en moins cloisonnée au domaine militaire car de plus en plus une affaire civile.

L'objectif de cette dissertation est double :

1. tout d'abord, nous analysons la force des attaques sur les implémentations physiques de systèmes de chiffrement symétrique, puis
2. nous proposons, en réponse, des contre-mesures raisonnées.

Cette mission est délicate, car il s'avère que les algorithmes utilisés en cryptographie, par exemple le chiffre symétrique, représente justement la classe d'algorithmes la plus facilement attaquable sur les canaux cachés. Par ailleurs, nous montrons que la sécurisation des implémentations est une tâche délicate, car même des attaques *a priori* non significatives, comme celle en « poids de Hamming », se trouve réussir, à cause d'effets de bords dévastateurs, car non anticipés. Il s'agit en l'occurrence de transitions « non-fonctionnelles », dites *glitches*. Nous détaillons l'origine de cette vulnérabilité, due à la fois à une dissymétrie de la porte OU-EXCLUSIF dans ses deux entrées et à une course temporelle de signaux. Il est intéressant de remarquer que cette même faille a permis de casser en 2006 une contre-mesure populaire aux attaques sur les canaux cachés, à savoir la logique WDDL [100].

Ayant pris conscience de la force des attaques sur les circuits, il s'agit de proposer des contre-mesures à la hauteur de l'enjeu sécuritaire. Étant donné que dans l'état de l'art des technologies industrielles permettant le calcul intensif (la filière CMOS n'a à jour pas de rival), il est impossible d'éliminer la consommation d'énergie, il va falloir composer avec les canaux cachés. Notre objectif est donc de faire en sorte que le canal caché (par exemple la consommation instantanée) mesurée par un attaquant ne fuit aucune information sur les données internes manipulées par le circuit. Notre étude consiste à symétriser les structures élémentaires de calcul, de telle sorte que tout calcul produise les mêmes symptômes physiques, du moins dans les limites imposées par les imperfections technologiques. Les portes logiques elles-mêmes sont donc garantes de la confidentialité des données qu'elles traitent. Grâce à une micro-architecture équilibrée

du point de vue géométrique, cette contrainte peut être implémentée, moyennant certains compromis d'ordre topologique. Effectivement, certaines symétries ne peuvent pas être respectées sans créer de court-circuits : c'est par exemple le cas de la symétrie axiale quand il s'agit de relier deux couples de points symétriques (A, B) & (B', A') de façon croisée, *i.e.* $A = A'$ & $B = B'$ électriquement parlant. Une étude sur la porte réalisant la fonction **NON-ET** se généralise à l'ensemble des primitives logiques utiles pour la réalisation de fonctions booléennes vectorielles arbitraires. Une bibliothèque complète de portes, appelée SecLib, est ainsi spécifiée et dessinée en technologie CMOS 130 nanomètres. Un effort particulier porte sur l'utilisabilité pratique de SecLib avec les outils d'assistance à la conception disponibles commercialement. De plus, la compatibilité de SecLib avec les bibliothèques dites « standards », fournies par les fondeurs de silicium, est une contrainte supplémentaire. Elle permet de mélanger les deux types de portes, sécurisés et normaux, dans un souci d'économie d'efforts de conception et de réutilisation d'éléments déjà validés par ailleurs. Un accélérateur cryptographique dédié au chiffrement multi-mode DES et triple-DES est conçu avec SecLib d'une part et une technique de placement-routage originale, appelée “duplication du dessin des masques” (“*backend duplication*” [102]) d'autre part. Cette réalisation démontre qu'une méthodologie de conception sécurisée est effectivement transférable vers le tissu industriel du semi-conducteur.

Contents

Résumé de la thèse en français	iii
Acknowledgments	xxix
Introduction	xxxix
1 Physical cryptography	1
1.1 Overview of the cryptography	1
1.1.1 Security needs	1
1.1.2 Security objects	1
1.2 The gray box model	2
1.2.1 Framework for the attacks considered in this PhD thesis	3
1.2.2 Side-channel analyzes provided with in this PhD thesis	3
1.3 Attack of flawed implementations	4
1.4 Common misconceptions	4
1.5 Audience	5
1.6 Personal contributions	5
2 SCA immanence	7
2.1 Structural attacks	7
2.2 Logical dissipation	7
2.2.1 Reversible logic	8
2.2.2 Conservative logic	12
2.2.3 Application to DES	20
2.2.4 Conclusion about reversible and conservative logics	22
2.3 Power attack model	22
2.3.1 Introduction to power attacks	22
2.3.2 Differential power analysis	23
2.3.3 DPA model	26
2.3.4 Electrical simulation of the DPA	29
2.3.5 Connections between DPA and conventional cryptanalysis	32
2.3.6 Conclusion of the power attack model	37
2.3.7 Illustration of DPA signal-to-noise ratio on histograms	38
2.4 Practical computing as security weaknesses	39
2.4.1 Integrability constraints	39
2.4.2 Hamming weight <i>versus</i> Hamming distance	41

3	DPA on DES	49
3.1	Secured crypto-processors design	49
3.1.1	Security target	49
3.1.2	Motivation for DES	49
3.2	A DES architecture operating in IP representation	50
3.2.1	Introduction on DES	50
3.2.2	DES datapath improvement thanks to a generalized pipelining	52
3.2.3	Optimal software / hardware partition to realize all DES variants	58
3.2.4	Performance evaluation of the proposed architecture	61
3.2.5	Comparison with other fast and versatile implementations of DES	63
3.2.6	Proposed architectures modifications for bit-slice P&R	65
3.2.7	Conclusion on the DES architecture	65
3.3	Fully combinatorial DES implementation	67
3.3.1	Combinatorial DES datapath	67
3.3.2	Security properties of the combinatorial DES	69
3.4	DES remarkable cryptological properties	70
3.4.1	DES weak keys and fixed points	73
3.4.2	DES semi-weak keys and anti-fixed points	74
3.5	DES remarkable SCA properties	77
3.5.1	Semi-weak keys	79
3.5.2	Weak keys	81
3.5.3	SCA properties generalization for arbitrary keys	84
3.6	Explanation for the differential traces using HW <i>vs</i> HD	90
3.6.1	Interpretation of the differential trace using HW	90
3.6.2	Interpretation of the differential trace using HD	92
3.6.3	Single <i>versus</i> multi-bit HW or HD selection functions	103
3.6.4	Conclusion: improvement of side-channels analyzes	106
3.7	Realization of the DPA on DES	107
3.7.1	Selection function for the DPA on the DES architecture of SecMat V1	107
3.7.2	DPA on traces integrals	108
4	Backend countermeasures against SCAs	113
4.1	Leaking no information	113
4.1.1	Information = distinguishability	113
4.1.2	Randomization	114
4.2	The secured library “SecLib”	114
4.2.1	Secured standard cells	115
4.2.2	CMOS structures for the secured cells	116
4.2.3	Use in a regular design flow	120
4.2.4	Performances	120
4.2.5	The “SecLib” library	121
4.2.6	Interconnect involvement in a circuit security	125
4.3	A place-and-route strategy for secured ASICs	128
4.3.1	Using differential logic to thwart SCA	130
4.3.2	The “backend duplication” method	131
4.3.3	The constraints required by the “backend duplication” method	132
4.3.4	“Backend duplication” method insertion into an existing design flow	133
4.3.5	Comparison of the “backend duplication” method with related works	134
4.3.6	Suitability of the “backend duplication” method with some logic styles	134
4.3.7	Backend triplication	136
4.3.8	Implementing a duplicated netlist	137
4.3.9	Conclusion on the “backend duplication”	139

4.3.10 Graphical illustrations of the “backend duplication” method	141
5 Conclusions	145
5.1 Summary of the dissertation	145
5.2 Perspectives	146
5.2.1 Open issues	146
5.2.2 Going further	146
A Attackees/attackers technical details	147
A.1 The SecMat circuits family	147
A.1.1 SecMat frontend	147
A.1.2 SecMat backend	151
A.2 The attack boards	151
A.3 The acquisition setup	152
A.3.1 Optimal power traces acquisition experimental conditions	152
A.3.2 The acquisition software	156
B Power Traces on the DES Co-Processor of SecMat V1	161
B.1 Hamming weight <i>vs</i> Hamming distance differential traces	162
B.2 DPA signal-to-noise ratios on DES	163
C Glossary	177

Acknowledgments

The work presented in this manuscript compiles the results of four years of research towards the definition of a backend-level solution to thwart power attacks on electronic integrated cryptosystems.

First of all, I am grateful to the members of the jury, to have accepted to review my work and to organize my public defense. Professor Francis JUTAND made me the invaluable honor of being the chairman of the jury. His broad knowledge of the information theory sector helped situating this work in a more global and quickly evolving context. Professors David NACCACHE and Jean-Jacques QUISQUATER thoroughly analyzed my manuscript and suggested relevant perspectives. Professor Marc Renaudin, through the work conducted under his supervision at TIMA, gave me insights related to power attacks realized on QDI (and other exotic logic styles) circuits. Doctor Pierre-Yvan LIARDET represented the industrial point of view on side-channel resistance: he motivated sound albeit integrable solutions against the side-channel leakage problem.

I could not have produced this document without the help of many folks.

To begin with, I wish to express all my sincere gratitude to Renaud PACALET, my supervisor. He has been able to make me take crucial decisions, while in the meantime leaving me a large space for innovation. This accounts for the large variety of skills that I learnt throughout this PhD. For instance, I had the rare opportunity to design two dedicated circuits in embedding full custom structures. It is from Renaud that got the conviction that there was both a scientific and industrial need for trusted computing platforms. Renaud's philosophy to find the best possible solutions before looking for acceptable trade-offs encouraged me: this idea gave birth to fundamental results, resisting the evolution of the state-of-the-art, which was reassuring me regarding the continuity of the work. As the leader of the **LabSoC** at Sophia Antipolis, Renaud also plunged the problematic of ASICs security into that of complex systems-on-chip. Many local contractual projects (especially with STMicroelectronics) added to the stimulation towards achieving a reliable DPA-proof design flow for trusted platforms.

Philippe MATHERAT taught me that the physics of the computation is a theory still at its infancy: the principal results remain to be discovered. He interested me on the links between *logical* and *physical* dissipation; this open question motivated my researches. Is the dissipation correlated to the difficulty of the computations? This problem is certainly a major theoretical challenge deserving of being tackled with in the next century!

Philippe HOOGVORST gave me inestimable insights about some advanced cryptographic issues. He has always been available to discuss some very accurate technical questions. Moreover, I owe him a lot of gratitude for his lessons in object-oriented coding and in software development technics.

All the other members of the Digital Electronic Group at the ENST (groupe "VLSI", later renamed "Systèmes Électroniques Numériques") also contributed to this work, everyone in its own manner. Yves MATHIEU, with its inexhaustible energy, encouraged me on an everyday basis. He also provided the adequate support when necessary: it is with its help that libraries were layout, that simulations were achieved, that complex designs were verified, and finally that circuits were timely taped-out. Jean-Luc DANGER got personally involved in the security

project, also trying to apply the results to other fields, such as reconfigurable computing and asynchronous logic. Jean PROVOST transferred me most of his valuable expertise in sub-micronic technology processes and in electrical simulation. Alexis POLTI and Lirida NAVINER DE BARROS's interest in embedded devices also influenced this work. I am also in debt from them for their pieces of advice when I was stuck on technical problems.

I also thank Karim BEN KALAIA for designing and debugging the attack boards. Without his help, no accurate side-channel measurements would have been possible. Karim provided and still provides useful guidance for the choice of the appropriate technical solution to the daily electric challenges.

The IT team, and especially the system administrators of the COMELEC laboratory, Arnaud LAURIOU and Frédéric PAUGET (staff of the CNRS LTCI – UMR 5141), installed the peripherals needed to control devices and to automate the traces acquisition campaigns. They guided the choice for the most adequate hardware and software resources to carry out the difficult task of building up a security evaluation platform. I especially thank Frédéric PAUGET for his proactivity in the choice for two critical computing resources: the 2-Terabytes SVN repository & SQL database server (`geant.enst.fr`) and the supercomputer (`genie.enst.fr`), tailored for the side-channel traces storage and analysis.

Laurent SAUVAGE actively contributed to the side-channel evaluation platform and solved many difficult issues related to side-channel measurements. I now wish that Laurent obtains brilliant results from the time he invested in the traces acquisition platform. Many thanks to Florent FLAMENT, for its exceptional involvement in the “SecMat” adventure, and for his highly valuable technical mastering and also for his precious sense of organization. Florent's work helped to structure the various developments around hardware attacks and counter-measures, and to capitalize on them. I am very proud to say that some brilliant results, especially those reported in chapter 3, were first suggested by Florent.

Many other people participated on aspects that are not accounted for in this manuscript. To cite a few: Mohamed EL HARHAR and Freddy ZANIN, for the design of the ancestor of SecMat, Hervé FINE and Nicolas PACHER for attacks on FPGA boards, Sumanta CHAUDHURI for the securization of asynchronous FPGAs, Virginia MARTÍN HERÍZ for her preliminary results on the DPA on AES, Johannes SCHMIDT for the implementation of novel attacks, Korinna LENZ for the attack against software implementation of DES running into an home-made fake smartcard based on an ATMEL ATmega processor.

The integration of this work on the circuits' security into a larger “Trusted Computing” GET¹-wide structuring project enlarged the scope for the hardware to the software. The contributions of Sophie COUDERT, Ronan KERYELL and Guillaume DUC, and of course of Renaud PACALET, were fundamental for the project creation.

Collaborations outside of the GET, mainly motivated by national projects, were also very fruitful. Many ideas originate from discussions with STMicroelectronics at Rousset, TIMA CIS and QLF groups at Grenoble, the multi-wafer project broker CMP at Grenoble, LIRMM at Montpellier, UCL at Louvain-la-Neuve and Oberthur at Nanterre. I also thank colleagues from the LIP6 of Université Pierre et Marie Curie for informal information exchange during the backend stages of the ASICs design.

Finally, I thank all my close friends and my family, that were sometimes wondering why I spent so much time working instead of caring of them. Thank you for your comprehension, and also for making me work smarter, not harder!

For the list to be really complete, I must thank the open source and/or free software developers and maintainers community. Many of them, through their advice and their software packages, helped me a lot, for a variety of tasks: computation, design automation, source code management, edition, typesetting and presentation. I especially thank the numerous volunteers of `bugs.gcc.org` and of `comp.lang.c++.`

¹The GET is the “Groupe des Écoles des Télécommunications”: <http://www.get-telecom.fr/>.

Introduction

This thesis is concerned with the security of electronic circuits against attacks on their implementation.

Cryptographic algorithms have traditionally been studied to withstand theoretical attacks, that are tacitly tantamount to *black-box* attacks. However, when these algorithms are implemented in a real devices, many other specific attacks become possible. With the advent of attacks on the physical implementations, some information can be diverted (*i.e.* extracted) from the circuits, or even altered (*i.e.* modified) by external means, which makes the security requirements more stringent. The new types of attacks are referred to as **observation** (or “**SCA**”, short for **Side-Channel Attack**) and **injection** (or “**FA**”, short for **Fault Attack**) attacks. These attacks, targeting the physical level, move the context from *black-box* to *gray-box*, because some internal variables can be partially read or written to [43]. The type of hardware that is vulnerable to SCA and FA is that where the power is supplied externally or easily accessible by an outsider: this includes contact and contactless smartcards, but also any mobile devices (handheld or not.) The exploitation of the electromagnetic field makes it possible to mount remote attacks on otherwise physically unattainable circuits. In this security context, the attacker model must be refined. In the introduction or the dissertation, the difference between the *logical* and the *physical* cryptanalysis is discussed.

Then an analysis of the SCAs [66] is carried out. They happen to be “structural attacks”, in the sense that are inherent to information processing. In particular, cryptographic algorithms, because of some Boolean properties of their architecture, are especially sensitive to SCAs. When applied on symmetrical ciphers, we show that SCAs are inescapable on ideal hardware without counter-measures, because the minimum information that leaks happens to be the full key itself. Then we show that SCAs are enhanced by the technological limitations, such as the limited datapath bitwidth, and that cryptographic algorithms increase the power of SCAs.

After that, we show how to avoid SCAs by leaking almost no information. And, given that information leakage is inescapable, how to build secured circuits? The DES case-study helps us to show how to evaluate the implementation dissipation via the power emanations. The architecture of a representative crypto-processor and the implementation issues are discussed. The co-processor security is evaluated w.r.t. logical and physical attacks.

The security flaws brought to the fore are then used to propose an unconditionally secure module. We discuss a method based on a technological balancing. Both gate-level design and CAD techniques are addressed: the SecLib secured library conveys a syndrome-free elementary computation kernel at the bit-level, while the “backend duplication” strategy allows for an efficient use of those gates in arbitrary complex netlists.

Finally, we conclude on the evaluation platform that has been set up, and we mention the open problems that remain to be tackled with.

Key words: Trusted computing, secured electronics, layout-level counter-measures, side-channel attacks, SPA/DPA, power analysis, physical cryptanalysis, topological constraints, geometrical symmetries.

Chapter 1

Physical cryptography

This chapter presents the vulnerabilities of electronic circuits arising when an attacker is able to acquire some physical emanations in addition to the public information. The attack scenario is thus augmented: “logical” security requirements remain, while “physical” security precautions add up. The objective of this chapter is to lay rational bases for the topics developed in the rest of the PhD dissertation. The two ideas that we promote are:

1. The need for rigorous threat assessment and associated proven counter-measures.
2. The fact that attacks are reproducible, and that their outcome is deterministic and inherent to both the algorithm and the *ad hoc* architecture used.

It is thus possible to unify logical and physical security requirements in a global security framework where defenses against both abstract and concrete attacks must be devised. This is the field of “physical cryptography”.

1.1 Overview of the cryptography

1.1.1 Security needs

From a user point of view, the cryptography is a set of solutions that can be assembled to provide solutions to some basic usages, such as:

- privacy,
- integrity,
- authentication,
- non-repudiation, *etc.*

1.1.2 Security objects

Similarly to every complex system, the cryptography can be divided into layers, that are illustrated in the table below along with an example of a desirable property;

Layer	Example of property
Boolean functions	Non-linearity
Algorithms	One-way
Protocols	Zero-knowledge
Applications	Authentication implemented

This structure allows an abstraction between the layers. For instance, the algorithms are most of time independent of the Boolean functions actually used, provided they feature a high degree of non-linearity. The protocols are themselves independent of the underlying algorithms. If for instance the one-wayness property is desired, then many algorithms can be used interchangeably. Finally, at the application level, the protocols are also seen as commodities. Application that rely on SSL/TLS (Secure Sockets Layer / Transport Layer Security) typically initiate a communication with an exchange of *ServerHello* and *ClientHello*, where the two parties agree for a common protocol.

In the framework of this thesis, we focus on the Boolean vectorial functions and algorithms. This selection is motivated by the facts that the attacks on the implementation usually target more specifically the lower layers and that the security of protocols and applications are more logical concerns.

The algorithms of concrete use can basically fall into three categories. Their resistance against cryptanalytic and brute-force (or exhaustive) attacks is respectively based on [60]:

1. Symmetrical encryption and message authentication codes (MACs):
 - Cryptanalysis: number of rounds, taylored substitution boxes,
 - Brute force: size of the key;
2. Hash:
 - Cryptanalysis: number of rounds, taylored substitution boxes,
 - Brute force: size of the hash-code;
3. Asymmetrical encryption:
 - Cryptanalysis: mathematical $\mathcal{NP}$ -complete problem,
 - Brute force: size of the key.

Symmetrical encryption, hash and MAC primitives are closely related. A block cipher, such as DES [71], can achieve the three. Similarly, a hash function, such as SHA [69], can do the same (the encryption algorithm derived from SHA has been baptized SHACAL.) There are no widespread MAC algorithm that is not based upon an encryption or an hash algorithm.

In the sequel, algorithms based on DES will be more specifically studied.

1.2 The gray box model

The general context for cryptographic attack is that of a black-box: internal variables are unknown. However, the algorithmic details are fully available to the attacker. This principle is known as Kerckhoff’s law. It encourages security-by-clarity design methodologies rather than security-by-obscurity. The security-by-obscurity is somehow similar to obfuscation: it makes the attack setup more difficult, but once this obstacle is overcome, it does not decrease its strength.

The concrete systems that are the targets of attacks are often very complex, which might confuse analysis when trying to identify their critical parts. The first goal of an attacker is indeed to locate the sub-system in charge of using confidential information. This sub-system is almost always a cryptographic block. Under the abovementioned “open system model”, the whole system reverse-engineering is supposed to be feasible; it is even assumed that the insulation of the cryptographic sub-system is easy as compared to its attack.

Paradoxally enough, the “architectural” complexity of a system is a burden to its security. Indeed, a complex system is structured in layers (similar to the OSI [110] stack), in which every

level of abstraction brings its own vulnerability. The security of a combination (even with embedded counter-measures) equals that of the weakest link.

In the context of attacks on the implementation, some internal variables, not always chosen nor known, can be read or written to [43]. These attacks are customarily referred to as side-channel attacks (SCAs), that divide into passive (or observation) and active (or fault) attacks.

Cryptanalysis consists in finding an exploitable bias, due to an unknown design flaw, that none of the up-to-now tests detected. In that sense, passive attacks on the implementations and cryptanalysis are much alike.

Active attacks are more powerful than passive ones: the DFA¹ of Piret and Quisquater [40] requires as few as two well-behaved faults to successfully attack AES, whereas DPAs [54] require in the best cases a few hundreds of traces. Active attacks are also more expensive, because their setup is not trivial [11]. Seldom gate-level counter-measures against FAs have been reported (refer for instance to [94] for some suggestions); One reason is that the reporting of a detected error is complex, and that this protection can itself be circumvented with faults. Therefore FA counter-measures are usually a matter of algorithms. Additionally, the FAs are very different in nature from SCAs and so are the counter-measures against them. Because the SCAs are already a very complex matter we decided not to consider FAs. Instead, we focus on observation attacks and associated counter-measures at the implementation-level.

1.2.1 Framework for the attacks considered in this PhD thesis

The security evaluations realized in this work meet the same standards as cryptanalysis: all the implementation information, such as algorithm and its layout, is supposed to be known to the attackers.

Moreover, the attacker can make an unrestricted use of the system. Typically, an unbounded number of calls to cryptographic functions is possible. The attacker can also tamper with the circuit that make up the system, for instance to access its power supply.

Furthermore, we make the hypothesis that the analysis laboratory itself (probes, synchronization signals, *etc.*) is embedded onto the chip.

In order to evaluate our work, custom ASICs have been designed. They allow for a full control over the algorithm implementation, and an accurate access to the synchronization and the physical parameters (such as the power, the EMI, the temperature, *etc.*) They also allow for a comparison between the measures and the simulations, which is essential from a scientific point of view when analyzing attacks results.

When dealing with counter-measures, **efficiency** is the first focus, irrespectively of **overhead**. Unless otherwise explicitly stated, this is the securization strategy we adopt in this thesis. Indeed, the goal of this work is to conclude regarding the efficiency of counter-measures when the maximum efforts have been spent. If an efficient (albeit expensive) solution is found, then it is relevant to optimize it without compromising its efficiency until its cost becomes acceptable for a given application. On the contrary, if no satisfying solution can be found whatever the cost, it would be useless to spend efforts in optimization. In this case, the very valuable result would be the conclusion that low level design counter-measures are unpractical.

1.2.2 Side-channel analyzes provided with in this PhD thesis

The analysis of side-channel information from power or electromagnetic traces requires a lot of care to properly *gather*, and then *understand* them. Some power traces are included in Appendix B, to avoid over-loading chapters with numerous graph data. We paid attention to annotate the graphs with the time expressed in clock periods and the amplitudes with the appropriate physical units. This enables a rigorous scientific interpretation of the data.

¹The technical acronyms are explained in the Appendix C.

The acquisition platform is described in Appendix A. It can be noticed that all the acquisitions are performed at 20 Gsample/s, which is in most cases much more than necessary to successfully attack an unprotected device (refer for instance to the attacks described in Sec. 3.7.2.) The attacked circuits were typically clocked at 32 or 66 MHz, which means that some hundreds of points are available for every clock period. However, this acquisition speed is not an over-kill from a scientific viewpoint, because it makes it possible to analyze fine events occurring within the clock period. Many results presented in the Chp. 3 would not have been obtained with a lower acquisition rate.

1.3 Attack of flawed implementations

A software bug can trivially make some attacks possible. For instance, a software bug may be exploited to read a sensitive memory area. Like in software, hardware flaws can fully compromise the security. For instance, if scan-chains or test-points are left unprotected, secrets could be disclosed. But in hardware, the consequences of design flaws can lead to more subtle leaks. In a design that lacks clock-tree buffers, the power consumption will be increased and so will probably be the information leaks. A dual-rail² circuit where the gates are balanced, but the routing is not, compromise the intended counter-measure based on symmetry. Ironically enough, those two problems were indeed encountered in our first ASIC (nicknamed SecMat V1) ...

All the results presented in this work take for granted that the studied implementations are correct, that is to say that they respect their specifications and that the specifications are sane (which is far from being obvious.)

1.4 Common misconceptions

When evaluating the strength of attacks targeting a piece of hardware, its security level is generally over-estimated. Let us compare the cost of a logical and a physical cryptanalysis on DES. For the sake of simplicity, the attack is chosen to be of *brute force* type in both cases. The cost function is taken equal to computation time:

- no space-time trade-off is used for the logical attack and
- the traces are accumulated as soon as they are captured in a DPA acquisition setup.

Then with the state-of-the-art hardware,

- $5\,369\,726 \approx 2^{22}$ simple-DES encryptions per second (assessed with the built-in self test “`openssl speed des`” [30])
- 1 acquisition per second with an oscilloscope, including the pre- and post-processing (trigger setting & saving the averaged results on the disks; for accurate figures, refer to Appendix B at page 161)

Thus, to keep the same level of security as DES (which is low w.r.t. current security standards), a DPA-proof hardware should withstand an attack using $2^{56-22} = 2^{34} \approx 17$ billion traces. This ratio is relevant in the respect that the rare resource is the minimum number of encryptions that an attack must realize to retrieve the full secret. This figure is much higher than what is usually considered in the literature. This cautionary note is to be taken all the more seriously as some attacks on unprotected implementations are practical with as few as a couple or tens of power traces. Such attacks are for instance reported in the article about ITA [32] when some

²Dual-rail circuits is a widely studied counter-measure against power analysis. It will be explained in deep details in the following chapters.

key bits are manipulated individually, which might happen in software implementations. On parallel architectures, we successfully retrieved six key bits (the ones at the input of the sbox #6) with as few as 28 averaged power traces, using a selection function not described in this thesis.

1.5 Audience

The target audience for this PhD thesis is mainly cryptographers and designers of secured embedded systems. Some results from the PhD has been published in journals or conferences proceedings. Some others are presented genuinely in this manuscript. Finally, some issues were not fully explored. They are presented as “open problems” or as “conjectures”, to be addressed in future works.

There does not exist one *definitive* reference covering the security of embedded designs against attacks on their implementation. The so-called “DPA book” [97] is scheduled for publication in December 2006 – January 2007. Most technical information about the state-of-the-art can be found on the proceedings of the **IACR workshops**. The newly created portal <http://www.sidechannelattacks.com/>, maintained by the **Reliable Computing Laboratory** of the Boston University (USA), aggregates the currently available academic information. Other sources of information can be found on-line, like the “Side Channel Cryptanalysis Lounge” from the network of excellence (NoE) ECRYPT [66] or the one term course in the “Real time and Embedded Systems” curriculum at Institut Eurécom (France) entitled “Security of Security Hardware” [73].

1.6 Personal contributions

This manuscript accounts for three personal contributions to the field of embedded systems security:

1. The formalization of SCAs, notably with the introduction of selection functions expressed as a sign function of a Boolean auto-correlation. Within this theoretical framework, the SCA attacks are shown to be all the more powerful as the underlying algorithm is strong against linear and differential cryptanalyses (Chapter 2.)
2. An accurate analysis of differential power traces, which enables the explanation of the peaks observed in a DPA attack (Chapter 3.)
3. A method, correct by design, to design leakage-proof circuits from a global point of view. The method takes advantages of geometrical symmetries to balance both gates and their interconnect. The method is proved viable, by the realization of a secured cryptoprocessor (Chapter 4.)

After this chapter 1, that introduced the security problematic and the objects involved in cryptography, the rest of the dissertation is structured as follows. Chapter 2 shows what are the intrinsic vulnerabilities of current cryptographic primitives. Chapter 3 is a practical example of the security flaws illustrated in the case-study of DES. Chapter 4 settles the basic conditions for a type of counter-measures. The nature and the modelization of leakage via the power is discussed. The discussion is technological, and mainly concerns digital CMOS implementations. Now, knowing that leaks are unavoidable, how can they be made balanced so that they convey no information about the data? Chapter 5 concludes the work and open perspectives towards better achievements in the field of electronics privacy.

Chapter 2

SCA immanence

This chapter shows that, whatever the symmetric encryption algorithm under attack, the analysis of side-channels allows an attack to extract the whole key. First, a logical dissipation model is presented. It proves that, unless dedicated counter-measures are taken, the minimal information dissipation is exactly equal to the key. Then, a model for a power attack, namely CARDIS 2004 [89] model, is presented. It shows that a side-channel attack is all the more powerful as the target algorithm is cryptographically strong. The model is shown to be applicable to real ciphers: it is extended from substitution/permutation block ciphers to Feistel-networks, such as DES. Finally, we conclude with some important remarks about practical implementation of the differential power analysis.

2.1 Structural attacks

This section motivates the researches carried out in this thesis about counter-measures against SCAs. Despite many efforts deployed to thwart those attacks since the seminal publication of Paul Kocher in 1998 [54], no fully satisfying solution has been proposed publicly. An explanation for this fact is analyzed in this section, based on two observations. First of all, the use of keyed bijections is shown to leak at least the information about the full key. This means that the optimal implementation in terms of leakage precisely reveals only the secrets, that are otherwise mixed with other non-necessary (technological) dissipations. Then, the conjunction of the cryptographic functions and implementation constraints are proven to especially increase the leakage. The very arguments used to build the security of functions, from a cryptanalysis viewpoint, are exploited to improve the SCAs. Thus, whatever the exact algorithm, SCAs can attack them because they feature some structural properties. As a consequence, SCAs are referred to as structural attacks [18] in this section.

2.2 Logical dissipation

The side-channel attacks are only possible if there exists a leak of any nature accompanying the computation. In a physical viewpoint, leaks can originate from two sources:

1. An irreversibility, which increases Boltzmann entropy of the computing system. Irreversibility occurs whenever it is impossible to reverse the computation because two or more former states would be equally suitable after a transition has occurred. For instance, both “5 + 7” and “4 + 8” evaluate to the same result, namely “12”. This phenomenon is called “logical dissipation” [76].

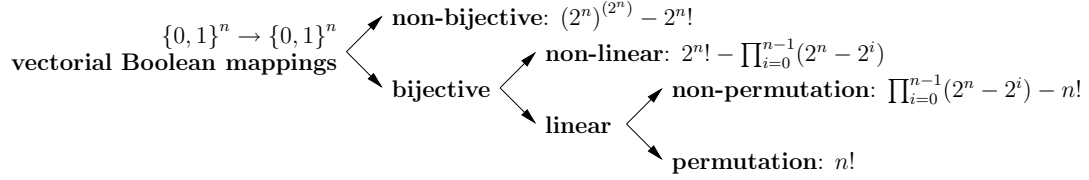


Figure 2.1: Taxonomy and counting of n -bit $\rightarrow n$ -bit vectorial Boolean mappings.

2. The dissipation caused by technological imperfections. Given the state-of-the-art technology, a theoretically reversible phenomenon happens to be irreversible in practice if not properly insulated from its environment. In CMOS technologies, the dissipation of a logical gate (such as a Boolean inverter) is caused by short-circuit currents and parasitic capacitances, as discussed later in Sec. 2.3. However, this leakage is due to imperfections in the implementations, and it is likely that they do not constitute fundamental limitations. They are thus commonly referred to as “technological dissipation”.

In this section, we assume that the “technological dissipation” can be reduced to null, and we focus on the computations that are realizable without any “logical” leaks. As a consequence only injections are of interest, because, knowing the image, a unique antecedents can be associated. For the sake of symmetry between the computation and the inversion of the computation, only injective invertible mappings such that the inversion is also injective are considered. Those functions are the bijections from $\{0, 1\}^n$ to $\{0, 1\}^n$. They are represented in Fig. 2.1. In this figure, the n -bit vectorial Boolean functions are partitioned into the relevant sets for cryptography.

The dissipation-free nature of reversible gates can alternatively be expressed in terms of entropy conservation. The entropy of a set of n independent random inputs x_i , $i \in [0, n[$ is equal to n bit. The probability of every input is $\mathcal{P}(x = x_{n-1} \cdots x_0) = \frac{1}{2^n}$. The entropy of the output of a reversible gate f is:

$$\begin{aligned}
 - \sum_{x=0}^{2^n-1} \mathcal{P}(f(x)) \log_2 \mathcal{P}(f(x)) &= - \sum_{x=0}^{2^n-1} \mathcal{P}(x) \log_2 \mathcal{P}(x) \quad // \text{Outputs reordering} \\
 &= - \sum_{x=0}^{2^n-1} \frac{1}{2^n} \log_2 \frac{1}{2^n} = +n \text{ bit} .
 \end{aligned}$$

This shows that the entropy variation after the reversible gate traversal is equal to zero.

2.2.1 Reversible logic

The implementation of reversible functions relies on the assembly of primitive gates to form a circuit. The decomposition of a function into the gates is a process called “logical synthesis”. After synthesis, a “netlist” of gates is proposed as one possible implementation of the function. The netlist can be seen as an oriented graph, where the nodes are the primitive gates and the arcs the interconnection nets. In the context of combinatorial logic, the graph is acyclic. In this netlist model, the “fork” (split of a net into other nets) must be represented as a gate, sometimes also referred to as the “fan-out” gate. A fan-out gate has more outputs than inputs. Thus, if the fan-out gate is used, there must exist “condensation” gates than have more inputs than outputs. Otherwise, the function produces more outputs than inputs, although it is specified to operate from $\{0, 1\}^n$ to $\{0, 1\}^n$. All the condensation gates are not reversible. A reversible synthesis of a reversible function does not utilize irreversible gates. Hence forks are forbidden

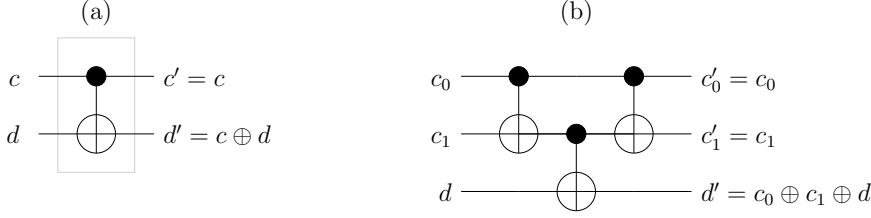


Figure 2.2: (a) The controlled inverter with one control c and (b) a construction to realize linear combinations of inputs.

in a reversible netlist. The advantage of this restriction is that any composition whatsoever of reversible gates is in turn reversible.

The one-input gates consist in the identity $a \mapsto a$ (the wire) and the inverter $a \mapsto \bar{a}$. With those primitives, only the permutations composed with bit-flips are realizable.

The non-trivial two-input two-output gates are variations around the exclusive-or, denoted XOR: modulo permutations and inversions of the wires, those gates compute: $(c', d') \doteq (c, c \oplus d)$. This operation is also known as the controlled inversion, or C-NOT. The usual symbol for the C-NOT is depicted in Fig. 2.2(a). The first input is the control, left unchanged, whereas the second input is the controlled data. As the C-NOT is linear, netlists of C-NOT can only give rise to linear functions. Actually, the netlists of C-NOT ensemble is exactly the linear functions. The sketch of the demonstration is given hereafter, for a mapping $\{a_i, i \in [0, n[\mapsto \{a'_i, i \in [0, n[\}$. The first component of the linear function a'_0 is equal to a linear combination of all the inputs. Figure 2.2(b) gives an idea for a possible netlist for a Boolean linear combination. Then, all the components but the last are recomputed as a function of $\{a'_0\} \cup \{a_i, i \in [1, n[\}$. The a'_0 can now serve as control, but is not used any more as a data. By repeating the same operation for all the outputs, the function is entirely mapped using only C-NOTs.

In a view to generate all the bijections, including the non-linear ones, Toffoli proposed his now eponymous gate. The gate has three inputs, and realizes the following mapping: $(c'_0, c'_1, d) = (c'_0, c'_1, (c_0 \cdot c_1) \oplus d)$. The schematic for the Toffoli gate $\boxed{T}$ is given in the last column of the Fig. 2.3. One possible internal implementation is given for every T_i in a “gray box”: it only intends to illustrates the gate’s function. Nevertheless, the implementation as such is not reversible, because of the use forks and of one AND gate (depicted with an ampersand “&” in Fig. 2.3.) But globally, the gate is reversible. It must thus be seen as a physical atom for the overall construction of a reversible netlist to remain valid.

Toffoli shows in the historical paper [107] that the inverter, the C-NOT and the Toffoli gate is an universal set for the reversible functions. Toffoli’s result is correct provided that the constant gate ‘0’ is added to the generating set. With this addition, the universal set can be represented by the generalized Toffoli gates $T_i, i \in [0, 3]$ represented in Fig. 2.3. One can notice that T_2 (resp. T_1) is also equal to T_3 (resp. T_2) where the first input is forced to ‘1’.

We provide here an original demonstration of the universality of the set $\{T_i, i \in [0, 3]\}$. The demonstration is on purpose informal because the point lays less in the result than in its consequences. It relies on the fact that any bijection can be seen as an element of σ_{2^n} , the symmetric group of 2^n elements, mapping the 2^n possible combinations of the n inputs into the 2^n possible combinations of the n outputs. The symmetric group is generated by the transpositions $\{\tau_{2^n}(i, j), i \in [0, 2^n[\text{ and } j \in [0, 2^n[\setminus \{i\}\}$ that swaps items i and j . The function’s inputs and outputs are implicitly assimilated with their corresponding integer value, i.e. $\{a_i, i \in [0, n[\} \Leftrightarrow \sum_{i=0}^{n-1} a_i \cdot 2^i$. The generalized Toffoli gate T_n is the transposition $\tau_{2^n}(2^n - 2, 2^n - 1)$. As a matter of fact, T_n leaves inputs $a_i, i > 0$ unchanged, because they are the control signals. The least significant bit (LSB) a_0 is toggled when $\prod_{i=1}^{n-1} a_i = 1$. As

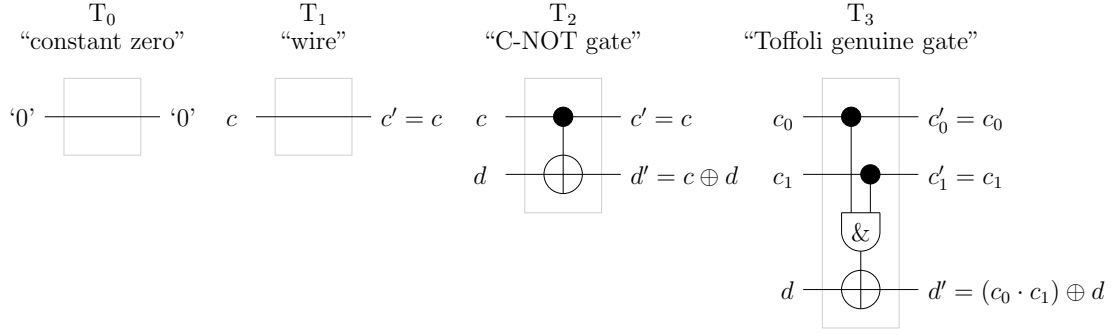


Figure 2.3: The first generalized Toffoli gates T_i . The genuine Toffoli gate [107] is T_3 .

a consequence, the n -input generalized Toffoli gate simply exchanges $1 \cdots 10 \Leftrightarrow 2^n - 2$ for $1 \cdots 11 \Leftrightarrow 2^n - 1$.

Now, the transpositions of σ_{2^n} are themselves generated from the sub-set $\{\tau_{2^n}(i, j), |i \oplus j| = 1\}$, where $|\cdot|$ denotes the Hamming weight. The other transpositions are then constructed as a composition of the “1-bit toggle” transpositions along a Grey path. As the variables can be sorted using renumbering, only “LSB toggle” transpositions, defined as $\{\tau_{2^n}(i, j), i \oplus j = 0 \cdots 01\}$ are eventually needed for a universal synthesis.

We introduce positive and negative control generalized Toffoli gates. In the original Toffoli gate, the control function f is defined as the condition when the data input toggles. This function is $f : c_i, i \in [0, n-1[\mapsto \prod_{i=0}^{n-1} c_i$. The function f can be made more general, by introducing the possibility that a control signal c_i be interpreted negatively. When this is the case, the corresponding input is marked with a “o” symbol (while regular or positive control inputs are identified by a “•” – not to be confused with the “solder dot” or the “fork” symbol, used for instance in Fig. 2.4(a).) An illustration of those gates is given in Fig. 2.4 for the case $n = 3$. The functionality of the positive/negative control variants of T_n is expressed by the mapping:

$$(a_0, a_1, \dots, a_{n-1}) \mapsto \underbrace{(a_0 \oplus \prod_{i=1}^{n-1} (\circ_i \oplus a_i))}_{\text{data signal}}, \underbrace{a_1, \dots, a_{n-1}}_{\text{control signals}}, \quad (2.1)$$

$$\text{where: } \begin{cases} \circ_i = \bullet_i = 1 \text{ if the control signal } i \text{ is negative, or equivalently,} \\ \circ_i \oplus a_i = \bullet_i \oplus \bar{a}_i = \bar{a}_i \text{ if } \circ_i \text{ or } a_i \text{ if } \bullet_i. \end{cases}$$

The synthesis of the positive/negative control variants can be achieved by the successive transformation of positive controls into negative ones. The process is illustrated by Fig. 2.5. The underlying idea is to use a $n - 1$ Toffoli gate to cause a double swap in the truth table, and then to cancel one swap using an n Toffoli gate. There only remains one swap (hence the composition yields an n Toffoli gate), but for another control function. By repeating this operation along a Grey path, all the positive/negative gates are realizable.

The partial conclusion is that all reversible functions can be synthesized in terms of T_1 , T_2 and T_3 , provided that T_i , for $i \in [4, n]$ are available.

In his article [107], Toffoli proposes a design scheme for $T_{i \geq 4}$ based on the idea depicted in Fig. 2.6. This construction demands to add one dummy input (a T_0 gate) and symmetrically one output (another T_0 gate) per primitive $n \rightarrow n + 1$ synthesis. This artifact is not considered a burden as long as it is free to add a “void” degree a freedom.

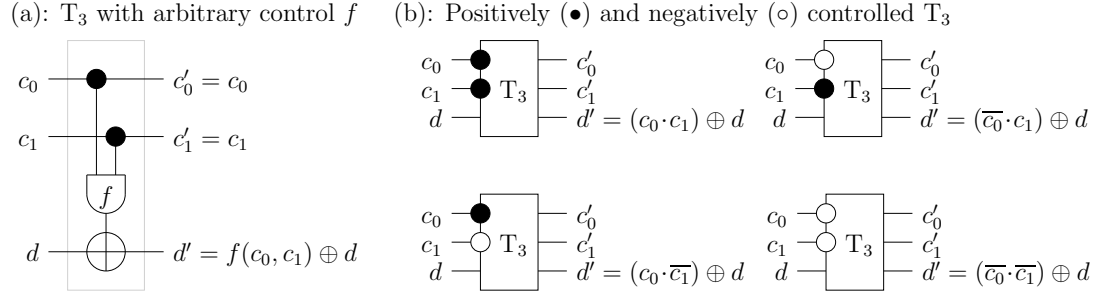


Figure 2.4: (a) Custom controlled T_3 gate by *a priori* arbitrary function $f: \{0, 1\}^{(3-1)} \rightarrow \{0, 1\}$.
(b) Illustration of four non-trivial positive/negative controlled T_3 variants defined in Eqn. (2.1).

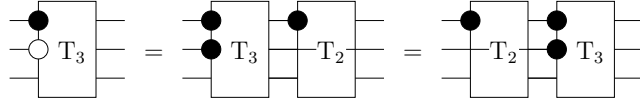


Figure 2.5: Synthesis of a positive/negative variant of a T_3 Toffoli gate.

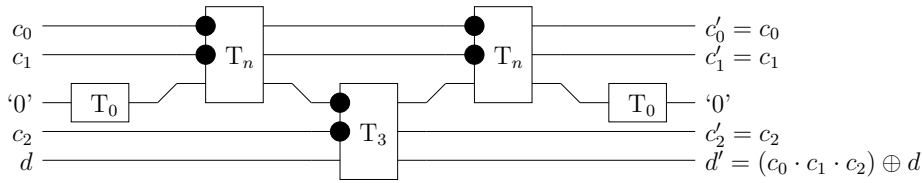


Figure 2.6: Synthesis of T_{n+1} from two T_n , one T_3 and two T_0 gates.

However, to remain in a strict $\{0, 1\}^n$ input space, Toffoli's construction cannot hold. There is actually a fundamental reason for the $n \rightarrow n + 1$ synthesis failure without the addition of a non-functional input. The T_{n+1} Toffoli gate corresponds to a transposition in $\sigma_{2^{n+1}}$, of signature -1 . When T_n is immersed into the bijections of $n+1$ bits, its signature becomes $(-1)^2 = +1$. It is thus impossible to generate T_{n+1} from σ_{2^n} gates. As a consequence, the universal reversible set that does not resort to constants, for $\{0, 1\}^n \rightarrow \{0, 1\}^n$ bijections, is: $\{T_i, i \in [1, n]\}$. This set does depend on the function dimensionality, because it holds $n - 1$ elements. Given the construction of the positive/negative control gates based on the collaboration of T_{n+1} and T_n (refer to Fig. 2.5), the set is proved to be minimal for netlists that do not alter the interface (*e.g.* add dummy input and output port couples.)

2.2.2 Conservative logic

The description of the reversible logic proves that a logically reversible function can be computed using a physical system. However, the physical system must be continuous [107]. It is desirable to study the classes of computations realizable with discrete physical systems, because digital electronic gates are typically represented as controlled interruptors manipulating bits. Typical discrete systems are a (classical) perfect gas or billiard balls moving in two directions on a table, as introduced by Fredkin *et al.* [35]. With those systems, it is mandatory that the number of molecules or of balls be conserved. This constraint defines the “conservative logic” [35].

The vectorial Boolean mappings that satisfy the conservation property are the reversible functions that preserve the Hamming weight. Indeed, one can code the presence (resp. the absence) of a particle by ‘1’ (resp. ‘0’). The inputs of a conservative bijective mapping can be partitioned according to their Hamming weight $p \in [0, n]$. These partitions must be preserved by the mapping, but inside of them, any permutation is legal. As a consequence, the number of conservative mapping is equal to: $\prod_{p=0}^n \binom{n}{p}!$, where $\binom{n}{p} \doteq \frac{n!}{(n-p)! \cdot p!}$ are the binomial coefficients.

In a similar approach as the one used for reversible logic, the minimal set of universal gates is researched. When $n \leq 2$, all the functions are linear. For $n = 1$, the only conservative function is the wire ($a \mapsto a$). For $n = 2$, the two conservative functions are the identity $((a_0, a_1) \mapsto (a_0, a_1))$ and the swap $((a_0, a_1) \mapsto (a_1, a_0))$. For $n > 2$, a non-linear primitive is required. Fredkin proposes a “controlled-swap” primitive, that realizes: $(c', d'_0, d'_1) = (c', d'_0 \cdot \bar{c}' \oplus d'_1 \cdot c', d'_1 \cdot \bar{c}' \oplus d'_0 \cdot c')$. The input c' is unchanged and controls the swap of data inputs d_0 and d_1 . As with reversible gates, the definition can be enlarged to encompass arbitrary function f control of two inputs, as illustrated in Fig. 2.7. Notice that the convention on the controlled swap polarity is the inverse of the one presented in [35].

It is worth noticing that Fredkin provides in [35] with two realizations of his gate in the billiard ball model, respectively nicknamed “R. Feynman and A. Ressler” and “N. Margolus”. The generalized Fredkin gate F_n introduced in this thesis can be trivially realized in the same model, by wrapping a F_{n-1} gate between a couple of switch / anti-switch (“R. Feynman and A. Ressler”) or interaction / anti-interaction (“N. Margolus”) gates fed with the extra control signal. This construction grows the gate linearly with the number of control signals added.

In F_n , the inputs a_0 and a_1 are swapped when all the controls are equal to ‘1’. The swapping leads to a difference when $a_0 \neq a_1$, *i.e.* when (a_0, a_1) is either equal to $(0, 1)$ or to $(1, 0)$. As a consequence, the n -input generalized Fredkin gate with positive controls simply swaps $1 \cdots 101$ and $1 \cdots 110$.

The conservative functions can be generated from the conservative transpositions. This property results from the fact that the set of conservative functions is isomorphic to the cartesian product: $\bigoplus_{p=0}^n \sigma_{\binom{n}{p}}$. This shows that the conservative functions can be generated by the composition of conservative transpositions $\tau_{\sigma_{2^n}}(i, j)$, with $|i| = |j| = p$, for all $p \in [0, n]$. The conservative transpositions can indeed be partitioned into $n + 1$ classes: if p belongs to $[0, n]$, then a conservative transposition of two items of Hamming weight p leaves unchanged other partitions of Hamming weight different from p .

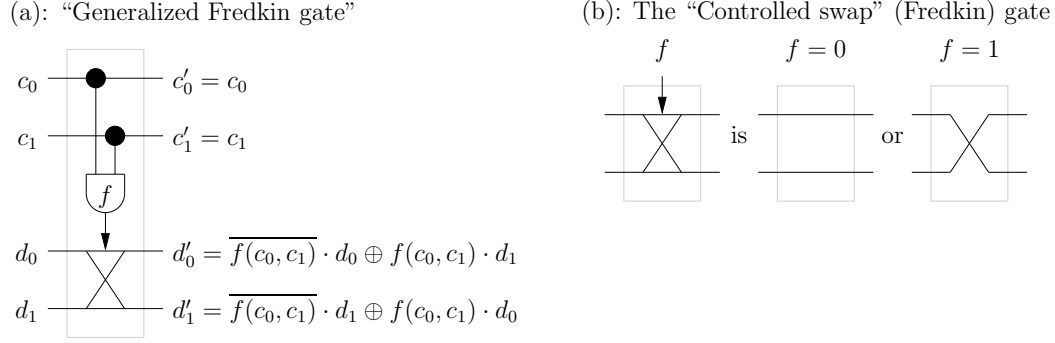


Figure 2.7: Fredkin generalized gate, illustrated in (a) with two control signals c_0 and c_1 , for which a control function f is evaluated, and pilots (b) a conditional swap.

Lemma 1 *Let i and j be two bit vectors satisfying $|i| = |j|$. Then $|i \oplus j|$ is even.*

Demonstration 1 (Proof of lemma 1) *Given two bits x and y , their exclusive-or can be evaluated with the following arithmetical equation: $x \oplus y = x + y - 2 \times x \cdot y$. Thus, if bit vectors i and j are equi-weighted, $|i \oplus j| = \sum_p (i \oplus j)_p = \sum_p i_p + \sum_p j_p - 2 \times \sum_p (i \cdot j)_p = 2 \times \sum_p (i_p - (i \cdot j)_p) \in 2\mathbb{N}$.*

The conservative transpositions can be generated from the subset of $\tau_{\sigma_{2n}}(i, j)$, for which $|i| = |j|$ and $|i \oplus j| = 2$. The demonstration is given in the form of the algorithm 1. For a list k , we use the following notations:

- the scalar $k.\text{back}()$ refers to the last element and
- the instruction $k.\text{push_back}(tmp)$ consists in appending the element tmp at the end of k .

An illustration of the algorithm 1 is provided in Tab. 2.1.

Algorithm 1 Computing all the conservative transpositions from the distance-2 transpositions.

```

1:  $k \leftarrow \{i\}$ ; {Initialization of a list (e.g. a C++ std::list) as a singleton.}
2: for  $p = 0$  to  $n - 1$  do {The  $p - 1$  first bits of  $k.\text{back}()$  and  $j$  are identical.}
3:   if  $i_p \neq j_p$  then
4:     for  $q = p + 1$  to  $n - 1$  do {The distance  $|(i_{n-1} \cdots i_{p+1}) \oplus (j_{n-1} \cdots j_{p+1})|$  is odd.}
5:       if  $i_q \neq j_q$  then
6:          $tmp \leftarrow k.\text{back}() \oplus 2^p \oplus 2^q$ ; {Moving two bits forward in the direction of  $j$ .}
7:          $k.\text{push\_back}(tmp)$ ; {Appending a new link in the chain  $k$ .}
8:       end if
9:     end for
10:     $p \leftarrow q$ ; {Advancing the outer loop counter}
11:  end if
12: end for {Post-condition:  $k.\text{back}() = j$ .}
13: return  $k$ ; {The walk from  $i$  to  $j$  through chain  $k$  is completed.}

```

The algorithm 1 allows the decomposition of the conservative transpositions according to:

$$\tau_{\sigma_{2n}}(i, j) = \begin{cases} Id & \text{if } k.\text{size}() = 1, \\ \bigcirc_{r=0}^{k.\text{size}()-2} \tau_{\sigma_{2n}}(k[r], k[r+1]) & \text{otherwise.} \end{cases}$$

Table 2.1: Illustration of the decomposition of $\tau_{\sigma_{2^n}}(i, j)$ for $n = 8$, initial value $i = i_7i_6i_5i_4i_3i_2i_1i_0 = 0\mathbf{x}62$ and final value $j = 0\mathbf{x}98$, where $|i| = |j| = 3$.

Input	Element of $\{0, 1\}^8$	Output
i	01100010	$k[0] \doteq i$ (Initialization)
	01101000	$k[1] \doteq k[0] \oplus 00001010$ ($p = 1, q = 3$)
	01011000	$k[2] \doteq k[1] \oplus 00110000$ ($p = 4, q = 5$)
j	10011000	$k[3] \doteq k[2] \oplus 11000000$ ($p = 6, q = 7$)

In the former equation, all the $\tau_{\sigma_{2^n}}(k[r], k[r+1])$ are distance-2 conservative transpositions, *i.e.* generalized Fredkin gates, which proves the decomposition proposition. More precisely, if p and q denote the indices of the differences between $k[r]$ and $k[r+1]$, then $\tau_{\sigma_{2^n}}(k[r], k[r+1])$ is the generalized Fredkin gate whose control function f (refer to Fig. 2.7) is: $x \mapsto \prod_{b \in [0, 2^n \setminus \{p, q\}]} (k[r]_b \oplus x_b) = \prod_{b \in [0, 2^n \setminus \{p, q\}]} (k[r+1]_b \oplus x_b)$. Notice that f , operating on the restriction $[0, 2^n \setminus \{p, q\}]$, can be rewritten thanks to the usual C-style binary operator `==` as: $x \mapsto (x == k[r]) = (x == k[r+1])$.

The previous demonstration is constructive, which proves the synthesizability of the studied functions. However, knowing that a solution exists, the actual synthesis can optimize the number of gates. For instance, all the 36 conservative bijections for an $n = 3$ dimensionality, defined in Tab. 2.3 can be decomposed using an exhaustive search of the netlists that instantiate the minimum number of Fredkin gates. Commercial synthesizers, using Boolean decomposition into (AND, OR) primitive gates prior to performing a technological mapping, cannot cope with such a synthesis. Although the conservative library shown in Tab. 2.2 can be parsed and understood properly by any legacy synthesizer, no technological mapping is possible in practice due to the lack of either AND or OR primitives. A dedicated synthesis tool must thus be written to achieve the Boolean decomposition with Fredkin gates. This tool has been devised at the ENST, and is fully operational for small values n of conservative bijections $\{0, 1\}^n \mapsto \{0, 1\}^n$ to map.

The optimally compact netlists are expressed in terms of the Fredkin gate (item 12) and of the 6 permutations (items 0, 13, 8, 22, 27 and 35). The former is represented in Tab. 2.4 and the latter are depicted as in Tab. 2.5. The netlists are provided with in Tab. 2.6. Notice that the netlists provided in this table are not unique.

Table 2.2: Liberty [46] library describing a conservative logic cells library (actually containing one unique “Fredkin” cell) for combinatorial vectorial Boolean functions mapping.

```

/**
 * Filename:      conservative.lib
 * Description:   A minimal conservative library based on the fredkin gate
 * Warning:      This library is logical, which means that it does not
 *               contain physical values (e.g. capacitance, timing, power)
 */

library( conservative )
{
  /** The type of the 'in' and 'out' ports */
  type( bus3 )
  {
    base_type: array;
    data_type: bit;
    bit_width: 3;
    bit_from: 0;
    bit_to: 2;
    downto: false;
  }

  /** The Fredkin gate */
  cell( fredkin )
  {
    bus( A )
    {
      bus_type: bus3;
      direction: input;
      pin( A[0] ) {}
      pin( A[1] ) {}
      pin( A[2] ) {}
    }
    bus( Y )
    {
      bus_type: bus3;
      direction: output;
      pin( Y[0] ) { function: "A[0]"; /** control signal */ }
      pin( Y[1] ) { function: "A[0] & A[1] + A[0]' & A[2]"; }
      pin( Y[2] ) { function: "A[0]' & A[1] + A[0] & A[2]"; }
    }
  }
}

```

#	Inputs
	(000) , (100) , (010) , (110) , (001) , (101) , (011) , (111)
#	Outputs
0	(000) , (100) , (010) , (110) , (001) , (101) , (011) , (111)
1	(000) , (100) , (001) , (110) , (010) , (101) , (011) , (111)
2	(000) , (010) , (100) , (110) , (001) , (101) , (011) , (111)
3	(000) , (010) , (001) , (110) , (100) , (101) , (011) , (111)
4	(000) , (001) , (100) , (110) , (010) , (101) , (011) , (111)
5	(000) , (001) , (010) , (110) , (100) , (101) , (011) , (111)
6	(000) , (100) , (010) , (110) , (001) , (011) , (101) , (111)
7	(000) , (100) , (001) , (110) , (010) , (011) , (101) , (111)
8	(000) , (010) , (100) , (110) , (001) , (011) , (101) , (111)
9	(000) , (010) , (001) , (110) , (100) , (011) , (101) , (111)
10	(000) , (001) , (100) , (110) , (010) , (011) , (101) , (111)
11	(000) , (001) , (010) , (110) , (100) , (011) , (101) , (111)
12	(000) , (100) , (010) , (101) , (001) , (110) , (011) , (111)
13	(000) , (100) , (001) , (101) , (010) , (110) , (011) , (111)
14	(000) , (010) , (100) , (101) , (001) , (110) , (011) , (111)
15	(000) , (010) , (001) , (101) , (100) , (110) , (011) , (111)
16	(000) , (001) , (100) , (101) , (010) , (110) , (011) , (111)
17	(000) , (001) , (010) , (101) , (100) , (110) , (011) , (111)
18	(000) , (100) , (010) , (101) , (001) , (011) , (110) , (111)
19	(000) , (100) , (001) , (101) , (010) , (011) , (110) , (111)
20	(000) , (010) , (100) , (101) , (001) , (011) , (110) , (111)
21	(000) , (010) , (001) , (101) , (100) , (011) , (110) , (111)
22	(000) , (001) , (100) , (101) , (010) , (011) , (110) , (111)
23	(000) , (001) , (010) , (101) , (100) , (011) , (110) , (111)
24	(000) , (100) , (010) , (011) , (001) , (110) , (101) , (111)
25	(000) , (100) , (001) , (011) , (010) , (110) , (101) , (111)
26	(000) , (010) , (100) , (011) , (001) , (110) , (101) , (111)
27	(000) , (010) , (001) , (011) , (100) , (110) , (101) , (111)
28	(000) , (001) , (100) , (011) , (010) , (110) , (101) , (111)
29	(000) , (001) , (010) , (011) , (100) , (110) , (101) , (111)
30	(000) , (100) , (010) , (011) , (001) , (101) , (110) , (111)
31	(000) , (100) , (001) , (011) , (010) , (101) , (110) , (111)
32	(000) , (010) , (100) , (011) , (001) , (101) , (110) , (111)
33	(000) , (010) , (001) , (011) , (100) , (101) , (110) , (111)
34	(000) , (001) , (100) , (011) , (010) , (101) , (110) , (111)
35	(000) , (001) , (010) , (011) , (100) , (101) , (110) , (111)

Table 2.3: The truth table of the $\binom{3}{0}! \cdot \binom{3}{1}! \cdot \binom{3}{2}! \cdot \binom{3}{3}! = 36$ conservative bijections of $\{0, 1\}^3$.

Index	12
Name	F
Diagram	

Table 2.4: The genuine Fredkin gate $F \doteq F_3$ schematic representation.

Index	0	13	8	22	27	35
Name	$P_0 = \text{Id}$	P_1	P_2	P_3	P_4	P_5
Diagram	$\begin{array}{c} a \text{ --- } a' \\ b \text{ --- } b' \\ c \text{ --- } c' \end{array}$	$\begin{array}{c} a \text{ --- } a' \\ b \text{ --- } b' \\ c \text{ --- } c' \end{array}$	$\begin{array}{c} a \text{ --- } a' \\ b \text{ --- } b' \\ c \text{ --- } c' \end{array}$	$\begin{array}{c} a \text{ --- } a' \\ b \text{ --- } b' \\ c \text{ --- } c' \end{array}$	$\begin{array}{c} a \text{ --- } a' \\ b \text{ --- } b' \\ c \text{ --- } c' \end{array}$	$\begin{array}{c} a \text{ --- } a' \\ b \text{ --- } b' \\ c \text{ --- } c' \end{array}$

Table 2.5: The six permutations (*aka* wires reordering or elements of σ_3) of $\{0, 1\}^3$.

#	Netlist	Diagram	Simplified diagram
0	P_0		
1	$P_1 \circ F$		
2	$P_3 \circ F \circ P_5$		
3	$P_3 \circ F \circ P_3 \circ F$		
4	$P_2 \circ F \circ P_5 \circ F$		
5	$P_2 \circ F \circ P_3$		
6	$P_3 \circ F \circ P_4$		
7	$P_3 \circ F \circ P_2 \circ F$		
8	P_2		
9	$P_4 \circ F$		
10	$F \circ P_3$		
11	$F \circ P_5 \circ F$		
12	F		
13	P_1		
14	$P_3 \circ F \circ P_5 \circ F$		
15	$P_3 \circ F \circ P_3$		
16	$P_2 \circ F \circ P_5$		
17	$P_2 \circ F \circ P_3 \circ F$		

Continued on next page ...

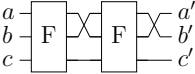
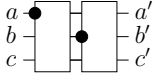
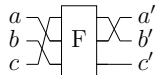
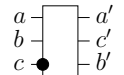
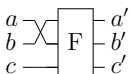
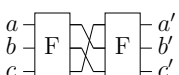
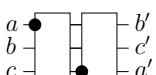
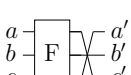
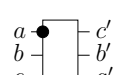
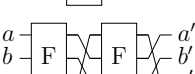
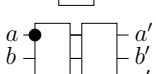
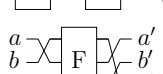
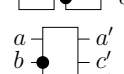
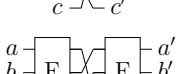
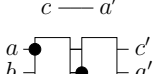
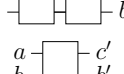
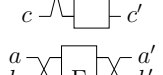
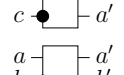
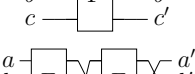
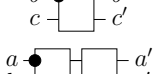
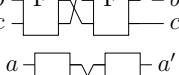
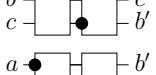
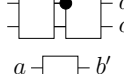
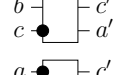
18	$P_2 \circ F \circ P_2 \circ F$		
19	$P_2 \circ F \circ P_4$		
20	$F \circ P_2$		
21	$F \circ P_4 \circ F$		
22	P_3		
23	$P_5 \circ F$		
24	$P_3 \circ F \circ P_4 \circ F$		
25	$P_3 \circ F \circ P_2$		
26	$P_2 \circ F$		
27	P_4		
28	$F \circ P_3 \circ F$		
29	$F \circ P_5$		
30	$P_2 \circ F \circ P_2$		
31	$P_2 \circ F \circ P_4 \circ F$		
32	$F \circ P_2 \circ F$		
33	$F \circ P_4$		
34	$P_3 \circ F$		
35	P_5		

Table 2.6: Minimal netlists of the 36 conservative gates of $\{0, 1\}^3$.

2.2.3 Application to DES

The revisited presentation of the reversible and conservative logic in Sec. 2.2.1 and 2.2.2 does not share the same objective as Toffoli and Fredkin's. They attempted to build a universal computational model resorting only to reversible or conservative primitives, adding whenever necessary dummy or duplicated inputs and corresponding outputs. Our goal is basically the same, except that it concentrates on a restricted class of functions, namely reversible and conservative functions. We proved that it is possible to synthesize them without adding neither non-functional inputs nor outputs. This constraint, that might appear artificial at this point, is indeed crucial in front of an active attacker. The dummy input or output can be exploited to inject faults, and thus open up on a potential vulnerability. Our approach is thus based on a precaution principle: without altering the problem's framework (n -bit to n -bit vectorial Boolean mappings), we have shown that a reversible or conservative synthesis is possible.

Given the nature of the underlying physics,

- *reversible* logic, adapted to *continuous* systems, fits to *analog* computation, whereas
- *conservative* logic, adapted to *discrete* systems, fits to *digital* computation.

The goal of this section is to apply the results from the conservative logic to a real-world algorithm, such as DES. If cryptographic computation often uses bijections, they are however usually not conservative. This problem can be alleviated thanks to a special data representation. For instance, the dual-rail encoding consists in using two wires (a_0, a_1) to encode one Boolean variable A . The couple (a_0, a_1) has constantly a unit Hamming weight with the following convention: $(a_0, a_1) \doteq (\bar{A}, A)$. The physical motivation is that every state of A ('0' or '1') conveys the same amount of energy, since they are encoded with globally indistinguishable states ('01' and '10'). Consequently, resorting to dual-rail, conservative logic can realize all bijections, conservative or not.

The Data Encryption Standard [71] cipher is built upon the following primitives: permutations, linear operations (typically XORs), and non-linear operations. The later are referred to as substitution boxes (or more shortly "sboxes"). The sboxes of DES are neither conservative nor reversible since they operate from $\{0, 1\}^6$ to $\{0, 1\}^4$. However, the input of every sbox is also forked (hence preserved unchanged), and the 4-bit output is the input of an XOR gate.

As a consequence, the non-linear and non-bijective logic of every substitution box can be seen as a function f controlling a Toffoli gate. This idea is explicited on Fig. 2.8(a), where, for the sake of clarity, the addition of the round key has been omitted. The input data are the bits 32, 1, 2, 3, 4 and 5 from a given register R . This input is kept verbatim in an output register called R' , that corresponds to the end of one DES round before the Left/Right half blocks swap. The controlled bits are bit 9 (but also 17, 23, and 31, computed in parallel by sbox #1) from another register L . The control function, denoted f in Fig. 2.8(a), cannot be obtained directly from the positive/negative Toffoli gates defined in Eqn. (2.1). Indeed, the sboxes output bits are not single min-terms of the control signals. Nevertheless, it is straightforward to instantiate enough chained T_{6+1} gates to end up with the *ad hoc* disjunctive normal form.

The same function can be realized in conservative logic using dual-rail encoding. The control wires are the true values of the *ad hoc* R bits, and the controlled data is the couple (L_0, L_1) . Indeed, if the control function is true, L is unchanged, otherwise it is inverted, which corresponds to the desired XOR functionality. The conservative sbox #1 first bit computation for DES is shown in Fig. 2.8(b).

For this reason, the synthesis remains reasonable, since the only gates required are:

- T_i for $i \in [1, 6 + 1]$ in reversible logic, or,
- F_i for $i \in [1, 6 + 2]$ in conservative logic,

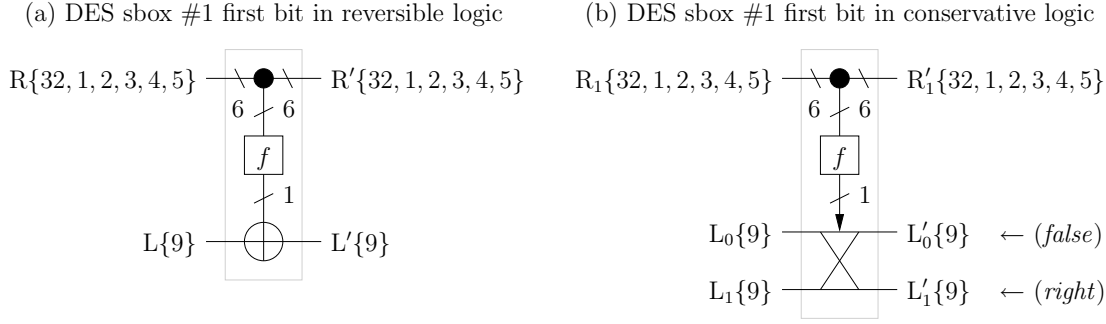


Figure 2.8: Computation of the first bit of the sbox #1 of DES, (a) in reversible logic and (b) in conservative logic.

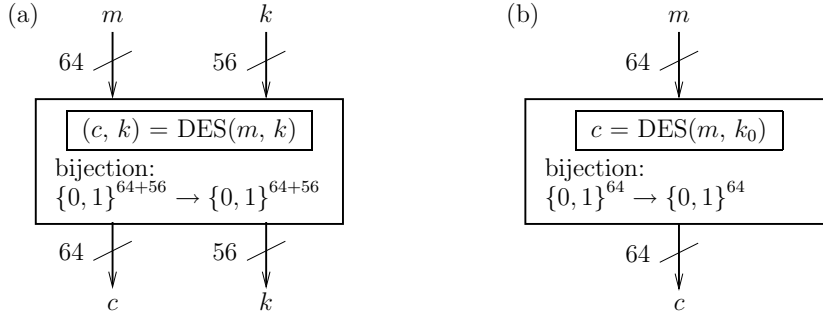


Figure 2.9: Abstract view of a symmetric cipher (here the Data Encryption Standard [71]), (a) generic in the key k or (b) personalized for a given 56-bit key k_0 .

because the input size of the DES substitution box is six bits.

Now, taking into account the fact that DES involves a 64-bit message and a 56-bit key, the full synthesis in reversible or conservative logic is feasible. The result in reversible logic is shown in Fig. 2.9(a). A similar schematic would be obtained in dual-rail logic for a conservative synthesis. The output of the reversible DES is a ciphertext and the key, that (in this very case) is returned unchanged, because of a specificity in the DES key schedule. After one encryption, only the ciphertext is disclosed; the key must be hidden from the user. It is thus the only information that is dissipated. In other terms, the sole bits that get erased are the very bits of the key.

To avoid this risk, the cipher could be specialized for one key, as depicted in Fig. 2.9(b). However, the latter case is not studied because it violates Kerckhoff's law. The secret is indeed dissolved into the implementation and will be vulnerable to a disclosure of the layout. A disclosure can typically happen if the circuit “repositories” (*i.e.* the source code to design, validate and build the fabrication masks) have leaked out on the internet, either by a mistake or by a malevolent intrusion. The layout must also be disclosed in the case of a legal suit, for instance if the plaintiff (the disguised attacker) claims a patent infringement about a pretext technological feature at the transistor level embedded in the layout.

The recent possibility to implement dynamic reconfiguration in FPGAs can be a work-around for a safe implementation of Fig. 2.9(b) that does not violate Kerckhoff's law [87].

2.2.4 Conclusion about reversible and conservative logics

The section 2.2 has proved that the realization of encryption algorithms is practical in both reversible and conservative logics. The number of unique primitive gates required for the synthesis of a product block cipher has been shown to be determined by the input size of the sboxes. Such implementations are immune to SCAs, the only problem being the key management. In fact, the secret key can be safely injected into the datapath, but its deletion after usage must be carefully devised to avoid its dissipation through a side-channel.

In the sequel, a regular CMOS [65] technology is used instead of reversible or conservative gates, for two reasons:

1. to our knowledge, no integrable and scalable Toffoli or Fredkin gates exist so far, and
2. synthesis tools are not available. The one that was written to map conservative logic functions (refer to Tab. 2.6) works in exponential time, and becomes unpractical starting from $n \geq 5$.

The use of CMOS logic opens up two security breaches, discussed hereafter. The imperfection of the gates (technological dissipation) as well as the erasure of bits of information (logical dissipation) can indeed be exploited.

The term cryptophthora (secret degradation) is sometimes used to express the degradation of secret key material resulting from side channel leakage. The rest of this chapter covers the topic of retrieving information correlated to a key, be it dissolved into the device or supplied externally, from a real device. First we show that the primitives used in algorithms make the attack outcome easier for the attacker. We focus on DPA attacks on non-linear parts of an algorithm, but we also mention that linear parts can enhance other attacks, such as the DFA. Then we show how concrete design constraints make the setup of attacks easy. We expose two constraints: the necessity to internally use some sub-keys, and the imperfection of standard logic gates.

2.3 Power attack model

Extended version of CARDIS'04 communication [89]

CMOS gates consume different amounts of power whether their output has a falling or a rising edge. Therefore the overall power consumption of a CMOS circuit leaks information about the activity of every single gate. This explains why, using differential power analysis (DPA), one can infer the value of specific nodes within a chip by monitoring its global power consumption only.

In this section, we model the information leakage in the framework used by conventional cryptanalysis. The information an attacker can gain is derived as the autocorrelation of the Hamming weight of the guessed value for the key. This model is validated by an exhaustive electrical simulation.

Our model proves that the DPA signal-to-noise ratio increases when the resistance of the substitution box against both linear and differential cryptanalyses increases.

This result shows that the better shielded against linear and differential cryptanalyses a block cipher is, the more vulnerable it is to side-channel attacks such as DPA.

2.3.1 Introduction to power attacks

Power attacks are side-channel attacks on cryptosystems implementing public or private key algorithms. They were first published by Kocher in 1998 [54]. Public key algorithms, like RSA, are vulnerable to simple power analysis (SPA), but can be efficiently secured by algorithmic

counter-measures [29, 88], like key and/or data blinding. Secret key algorithms, such as DES or AES, consist in the repetition of several rounds, and are thus threatened by the differential power analysis (DPA.)

DPA can attack on either the first or the last round of an algorithm and requires the knowledge of either the cleartext or the ciphertext. The side-channel exploited is the difference between the power consumed by a single gate when its output rises or falls.

Similar attacks take advantage of other types of leakage that disclose information about the internal computation. For instance, the correlation power analysis (CPA [111, 112]) monitors the activity of a register: the attack exploits the fact that in CMOS logic, a gate only dissipates energy when it changes states. CPA, unlike DPA, can be modeled with the assumption that the energy dissipation is independent on the gate (either rising or falling) edge. Those attacks can also be conducted by recording a different physical quantity than the power consumption, like the electromagnetic field [37].

The rest of this section is organized as follows: Sec. 2.3.2 explains the principle of the DPA attack. In Sec. 2.3.3, we present a theoretical model for the DPA. The model is validated against exhaustive electrical simulations in Sec. 2.3.4. In Sec. 2.3.5, some results prove that the better shielded against linear cryptanalysis a block cipher is, the more vulnerable it is to side-channel attacks such as DPA.

2.3.2 Differential power analysis (DPA)

2.3.2.1 Measuring the consumption bias of a CMOS inverter

The schematic depicted on Fig. 2.10(a) has been implemented using discrete NATIONAL SEMICONDUCTOR CD4007M transistors to measure the instantaneous current drawn from the power source VDD and sent back to the ground VSS. The two resistors R_N and R_P are not functional. They have been added to ensure that the voltage drops between:

1. VSS and point **A** and
2. VDD and point **B**

reveal respectively the currents $I(\text{VSS})$ and $I(\text{VDD})$. According to Ohm's law:

1. $I(\text{VSS}) = (V(\mathbf{A}) - V(\text{VSS})) / R_N$ and
2. $I(\text{VDD}) = (V(\text{VDD}) - V(\mathbf{B})) / R_P$.

In the context of a power analysis, an attacker also introduces a resistor R_N or R_P (referred to as a “*spying resistor*”) outside of the circuit under attack. The side-channel acquired by this mean can be equivalently seen as:

- a voltage drop (expressed in volts), or
- a current (expressed in ampers), or
- a power (product between the constant value of $(\text{VDD} - \text{VSS})$ and the current, expressed in watts.)

In the sequel, the three units are used interchangeably to quantify the leaks.

When the inverter evaluates to false ($S = \text{VSS}$), the N transistor is conducting whereas the P is blocked. The contrary happens when the inverter evaluates to true ($S = \text{VDD}$.) The inverter's transistors are thus modeled by interrupters, as in Fig. 2.10 (b) and (c). Fig. 2.11 shows that the current $I(\text{VDD})$ flowing through resistor R_P is the sum of:

- a short-circuit current, I_{short} , whose intensity is independent of the edge (*falling* or *rising*) of the output S of the inverter and of

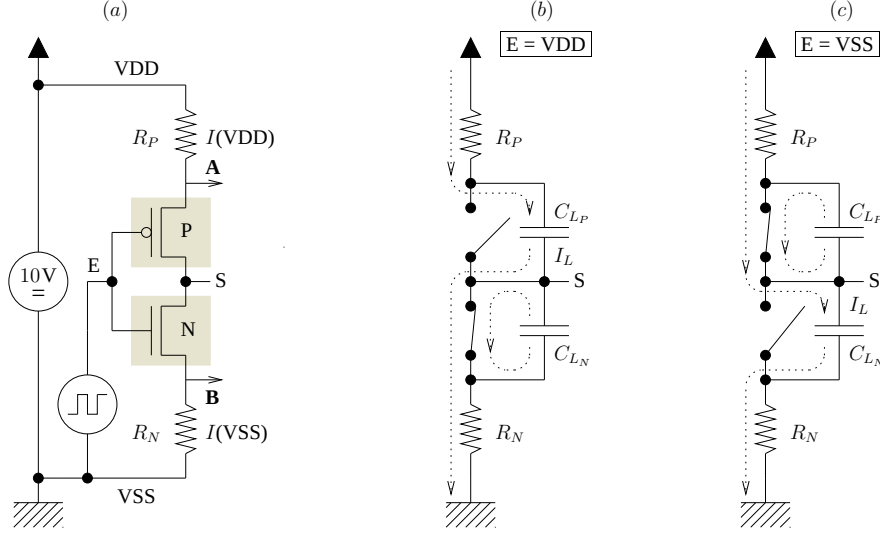


Figure 2.10: (a) Experimental setup used to measure the currents $I(VDD)$ and $I(VSS)$ when the output S of a CMOS inverter switches. The currents flows are illustrated in (b) and (c) for respectively a falling and a rising edge of the inverter's output S.

- a current I_L , loading a charging capacitance C_L and observed only when S rises from VSS to VDD (Fig. 2.10(c)), because, otherwise, C_L discharges through R_N only (Fig. 2.10(b)). The capacitance C_L models both the gate output capacitance, linked to the gate fanout and to the routing wires, as well as the parasitic capacitances.

The current $I(VDD)$ depends on the edge (rising or falling) of S. We denote:

- $I_{\downarrow} = I_{short}$ the current observed upon a $VDD \rightarrow VSS$ edge and
- $I_{\uparrow} = I_{short} + I_L$ the one observed upon a $VSS \rightarrow VDD$ edge.

2.3.2.2 Principle of the DPA attack

The analysis of the instantaneous power consumption can leak the type of operations being performed. For instance, Fig. 2.12 shows that the power consumption of a DES operator indicates the beginning of every encipherment. The retrieval of information from a single power trace is referred to as SPA.

Moreover, a more precise analysis can insulate the activity of a single gate, because:

- the instantaneous consumption of the circuit is the sum of all individual consumptions,
- each gate draws a different intensity ($I_{\uparrow}$ or $I_{\downarrow}$) according to its output edge, as shown in the previous example of the CMOS inverter.

The DPA attacks proceed in two phases. First, a large number of power consumption traces for different plaintexts¹ are recorded. Those traces contain the information about the type of edge (via a $I_{\uparrow}$ or $I_{\downarrow}$ contribution) of each gate in the design.

The second step consists in extracting this information from the traces $T_x(t)$. In the historical DPA [54], Kocher suggests to partition the traces according to the value of a particular

¹The plaintexts need not be known: the DPA can be a ciphertext-only attack. It can also be a plaintext-only attack, chosen or not [39].

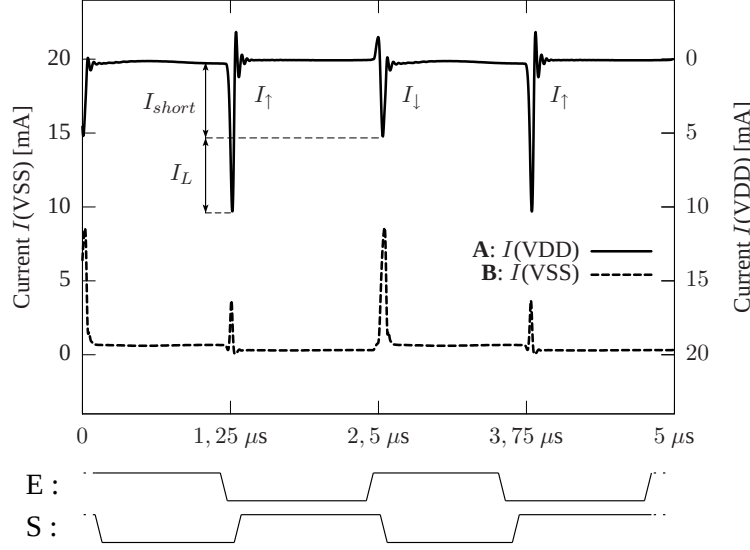


Figure 2.11: Measures of $I(VDD)$ et $I(VSS)$ of the inverter of Fig. 2.10(a) acquired by an oscilloscope.

bit i of the algorithm, which (hopefully) corresponds to a particular node in the netlist. One partition, S_0 , gathers the traces $T_x(t)$, where $i = 1$, expected to contain an $I_\uparrow$ contribution, whereas the other, S_1 , gathers the traces where $i = 0$. Thus the “differential trace”, computed as:

$$\frac{1}{\#S_0} \sum_{T_x \in S_0} T_x(t) - \frac{1}{\#S_1} \sum_{T_x \in S_1} T_x(t),$$

reveals the $I_\uparrow - I_\downarrow$ power consumptions of the target gate i . This *modus operandi* can be used as an oracle to validate or invalidate an assumption. The DPA attack consists in testing whether the differential trace feature a singularity (peak) when analyzing the consumption of a gate i whose unknown state is guessed by making an hypothesis on a secret (typically a part of a round key.) When the hypothesis on the key is correct, the differential trace is expected to feature a peak, resulting from the accumulation in a coherent manner of the $I_\uparrow - I_\downarrow$ information extracted from the power traces. More precisely, the peak is expected around the date t_S when the gate switches.

Refinements on this attack have been put forward [61]. The idea is to take into account that, in CMOS technologies, a gate only dissipates power when its output switches. The traces are thus partitioned into three sets. In addition to Kocher S_0 and S_1 sets, the S_2 set contains the traces with no or little dissipated power. Only traces from the sets S_0 and S_1 are used to compute the differential traces. For the sake of clarity, and to prepare for the presentation of our DPA model, we prefer not to present DPA in terms of traces partitioning but rather in terms of traces weighting. This allows us to reformulate the definition of the differential traces as a weighted accumulation of power traces, the weights being $+1$, -1 and 0 for traces belonging to sets S_0 , S_1 and S_2 .

2.3.2.3 Ghost peaks in differential traces

It has been reported in [111] that “ghost” peaks also appear in differential traces computed with a wrong assumption of the key. We explain in the next section that those secondary peaks can be as high as the peak for the correct key and we provide a theoretical way to compute their relative amplitude.

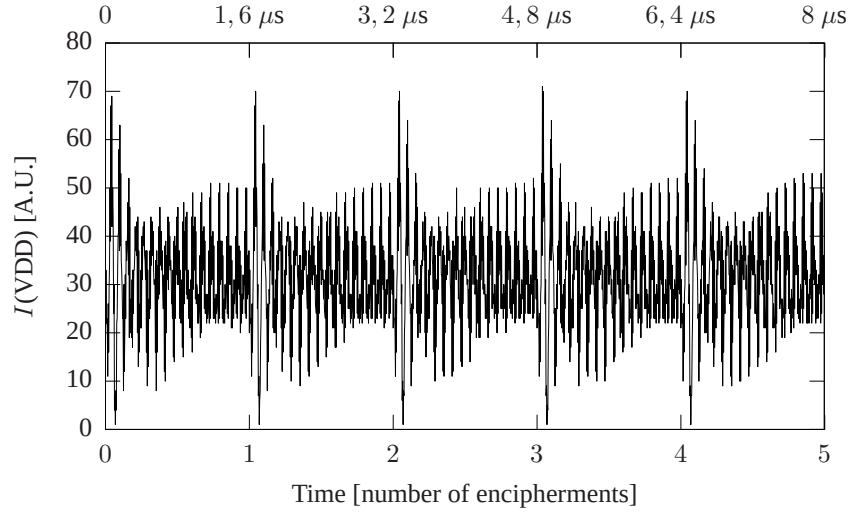


Figure 2.12: Power consumed by five DES encipherments (programmed in an FPGA).

2.3.3 DPA model

2.3.3.1 Framework for DPA

Model general setup. The DPA model we describe is applicable to hardware implementations of private key product ciphers. The algorithm consists in the repetition of some rounds, the first or the last one being attackable by DPA. Without any loss of generality, we focus on an attack on the last round. Fig. 2.14 shows the typical dataflow of one round: the “plaintext” corresponds to the intermediate message produced by the penultimate round which is mixed in the last round with the “key” to produce the “ciphertext”. The last round features one non-linear function (called S-Box) and one linear function (in our case a bit-wise XOR-ing) with some bits of the round key k . Given a known value x and an unknown but constant key k , the value y of all the target bits $i \in [0, q[$ under investigation is derived as:

$$y = F(x \oplus k). \quad (2.2)$$

In the original DPA [54], the value of each bit i of y is used to partition the traces so as to build differential traces. In other words, the “selection function” D introduced by Kocher is the projection of Eqn. (2.2) on i . In our model, the whole value of y is used to weight the traces in a view to obtain one differential trace.

As explained below, the selection function of Eqn. (2.2) applies to both DES and AES. Notice that side-channel attacks against block ciphers that do not use sboxes, such as RC5, are still an open topic.

DES. Fig. 2.13 represents a simplified dataflow of the last round of DES: the permutations and the expansion are left apart since the attacker can work around them. The guess on the bit i (belonging to the right part of the round 15 output) comes down to a guess on a bit of the output y of the S-Box, since C_L is known to the attacker. DPA on DES is therefore a particular case of Fig. 2.14, where $F = S$ is the direct S-Box: $K^p \rightarrow K^q$ with $p = 6$ and $q = 4$ (we denote $K = (\{0, 1\}, \oplus, \cdot)$ the field with two elements.)

AES. The schematic of Fig. 2.14 comes in a direct line from the structure of the last round of AES, with $F = S^{-1} = \text{InvSubBytes}$ and $p = q = 8$. This structure is referred to as

SPN, for Substitution – Permutation Networks. With Feistel networks, it is the second kind of symmetrical block encryption structures to be encountered.

Fig. 2.13 schematic actually also applies to any Feistel cipher with constant S-Boxes, in which the attacked bit belongs to the right part of the penultimate round.

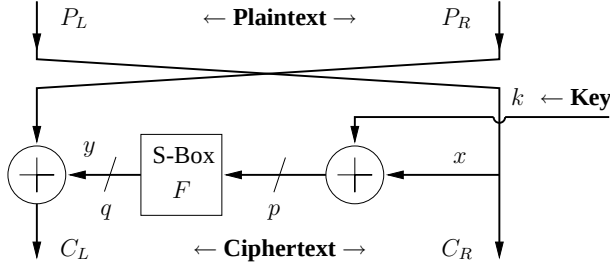


Figure 2.13: Simplified DES cipher flow showing a single S-Box out of eight.

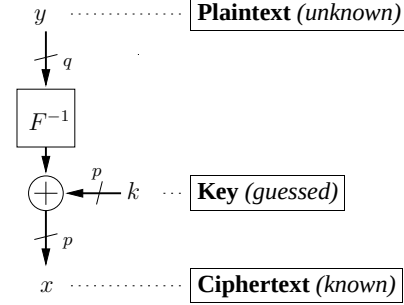


Figure 2.14: Schematic DPA setup on an SPN structure.

2.3.3.2 Noise sources occurring during DPA

There are various sources of noise when doing a DPA:

- N1. The activity of the rest of the circuit. This noise can be lowered by the accumulation of many independent traces. Noise spectral power vanishes as the inverse square root of the number of traces recorded.
- N2. The jitter on the attacked gate. Depending on the delays in the lines and the type of edges, the switching of a gate output can happen at different dates, which leads to a loss of coherence of the trace accumulation. This is negligible for gates directly fed by registers, as their inputs are perfectly synchronized.
- N3. S-Boxes themselves introduce their own bias. Measured traces slightly match the activity deduced from the computation of one plaintext bit $y_i = F_i(k \oplus \cdot)$, as described by Eqn. (2.2), even if the assumption on k turns out to be wrong. Although substitution box bits are designed to be independent from one another so as to block linear cryptanalysis, DPA *modus operandi* artificially introduces an inter-bit correlation. The plaintext bits y are computed all together which introduces an artificial correlation between them. This notion of S-Box “intrinsic noise” is investigated in the next section. It is also called “ghost peaks” in [111] and “algorithmic noise” in [61].

2.3.3.3 DPA intrinsic noise

2.3.3.3.1 The “DPA signal”: a model for the differential traces peak amplitude.

In this section we assume that the noise sources N1 and N2 are low enough. Under this condition, the DPA makes it possible to insulate the power consumption ($I_{\uparrow}$ or $I_{\downarrow}$) resulting from the activity of one single bit i : this manifests as a peak in the differential trace when the key is guessed right.

We propose to model the amplitude of the peak observed in the differential trace as a “DPA signal” that is built by an accumulation of scores. Given one ciphertext x , this score is:

- $\boxed{+1}$ if the value $F_i(k \oplus x)$ inferred for the bit i by the selection function (Eqn. 2.2) is the same as the actual $F_i(k_0 \oplus x)$ for the correct key k_0 ,

$\boxed{-1}$ otherwise. As the recomputed bit value is false, the trace is considered to provide a $I_{\uparrow}$ power consumption contribution whereas the actual contribution is $I_{\downarrow}$, or vice-versa. Thus, instead of accumulating coherently $I_{\uparrow} - I_{\downarrow}$ to the differential traces, the opposite $I_{\downarrow} - I_{\uparrow}$ will be added, thus reducing the score coherence.

The scores are accumulated over many encipherments. Asymptotically, the accumulation is done for all the ciphertexts x .

As already mentioned when discussing the noise source N3, all the q bits of $y = F(k \oplus x)$ are guessed at the same time. As they take their values simultaneously, it is impossible to test the $y_i = F_i(k \oplus x)$ independently.

The “DPA signal” is thus the accumulation over all the ciphertexts x of the score obtained by a bit i of the plaintext against the q bits of actual plaintext. As a result, the DPA signal is built from the correlation of plaintext bit i for the trial key k with plaintext bit j for the actual key k_0 :

$$\begin{aligned} \mathbf{Corr}(k_0, k; \mathbb{1}_i, \mathbb{1}_j), \quad & \text{where: } \forall k_0, k \in K^p, \forall a, b \in K^q, \\ \mathbf{Corr}(k_0, k; a, b) & \doteq \frac{1}{2^p} \sum_x (-1)^{\langle a | F(x \oplus k) \rangle \oplus \langle b | F(x \oplus k_0) \rangle} \end{aligned} \quad (2.3)$$

as [29]:

$$\mathbf{DPA}(k_0, k; \mathbb{1}_i) = \sum_{j=0}^{q-1} \mathbf{Corr}(k_0, k; \mathbb{1}_i, \mathbb{1}_j). \quad (2.4)$$

Moreover, as it is easy to prove that:

$$\mathbf{Corr}(k_0, k; a, b) = \mathbf{Corr}(k_0 \oplus k, 0; a, b),$$

the correlation is independent of the actual key k_0 . This means that there are no *weak keys* as for the differential power attack. The relevant parameter is the difference $k_0 \oplus k$ between the actual key and the trial key. We simply denote this difference k , as if the actual key was 0. The correlation (Eqn. 2.3) is rewritten $\mathbf{Corr}(k; \mathbb{1}_i, \mathbb{1}_j)$. The “DPA signal” is rewritten accordingly: $\mathbf{DPA}(k_0, k; \mathbb{1}_i) = \mathbf{DPA}(k; \mathbb{1}_i)$. The correlation takes its values in $[-1, +1]$ and equals $+1$ if the guess on the key is correct (*i.e.* $k = 0$) and $i = j$.

2.3.3.3.2 The “ghost peaks”. When recomputing one bit of the plaintext from the ciphertext and a wrongly guessed key, the value can, by chance, match the actual value. If it happens too often, the guessed key might be hard to distinguish from the actual key.

For instance, in the case of DES S-Box #3, there exists one wrong key that leads to a “DPA signal” (Eqn. 2.4) as high as the one for the correct key: it occurs when the bits 0 or 3 of the S-Box output are guessed.

Those secondary peaks make it difficult to interpret the differential traces: they make up an artificial noise that was reported as “ghost peaks” [111].

DPA *modus operandi* justification

In this section we assume that the S-Box F is balanced and that the attacker found the correct key (*i.e.* $k = 0$). If the partitioning test is done according to the value of $\langle a | F(x) \rangle$, where a belongs to K^q , (*e.g.* if $a = \mathbb{1}_i$, the sole bit i is used to partition the traces), the attacker

computes the following DPA signal:

$$\begin{aligned}
\mathbf{DPA}(0; a) &= \sum_{j=0}^{q-1} \frac{1}{2^p} \sum_x (-1)^{\langle \mathbb{1}_j | F(x) \rangle \oplus \langle a | F(x) \rangle} = \frac{1}{2^p} \sum_{j=0}^{q-1} \sum_x (-1)^{\langle \mathbb{1}_j \oplus a | F(x) \rangle} \\
&= \frac{1}{2^p} \sum_{j=0}^{q-1} 2^p \delta(\mathbb{1}_j \oplus a) \quad (\text{because } F \text{ is balanced}) \\
&= \begin{cases} 1 & \text{if there exists one } i \in [0, q[\text{ such as } a = \mathbb{1}_i, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

It shows that DPA exhibits a non-zero signal *if and only if* (iff) the partition is made on one of the q plaintext bits. In this case, the DPA signal is maximum (+1).

2.3.3.3.3 A new *modus operandi* for DPA. The traditional *modus operandi* for the DPA is to compute the differential traces for testing the value of the q bits of F output. However, the q differential traces are not independent, because the each predicted bit i is matched against all the actual plaintext $F(k_0 \oplus \cdot)$. For this reason we consider the sum of the q differential traces. The DPA signal associated is:

$$\mathbf{DPA}(k) \doteq \sum_{i=0}^{q-1} \mathbf{DPA}(k; \mathbb{1}_i) = \frac{1}{2^p} \left(\sum_{i=0}^{q-1} (-1)^{F_i} \otimes \sum_{j=0}^{q-1} (-1)^{F_j} \right) (k). \quad (2.5)$$

As an auto-correlation, the signal $\mathbf{DPA}(k)$ is maximum in absolute value in $k = 0$, *i.e.* when the attacker guess on the key is correct. This remark is the consequence of the Cauchy-Schwarz theorem applied on the pseudo-Boolean function $k \mapsto \mathbf{DPA}(k)$ [19].

The method to compute the differential trace can be reformulated. Let k be the key being evaluated. For every ciphertext x , the power trace is weighted by $W(x, k)$, the centered Hamming weight of the recomputed plaintext (Eqn. (2.2)):

$$W(x, k) \doteq \sum_{i=0}^{q-1} F_i(x \oplus k) - q/2 = \frac{1}{2} \sum_{i=0}^{q-1} (-1)^{F_i(x \oplus k)}. \quad (2.6)$$

The weighted power traces are accumulated to yield the differential trace.

2.3.4 Electrical simulation of the DPA

The DPA attack is simulated at the electrical level in order to validate our DPA signal model (Eqn. 2.5).

We find that, given the long time required by electrical simulations, a 6×4 S-Box like one S-Box of DES cannot be simulated for all the plaintext transitions. Instead of limiting ourselves to a subset of the possible messages, like in [104], we choose to simulate a simpler cryptographic operator. The cipher used is the one shown in Fig. 2.14, with Serpent [85] S-Box #0 ($p = q = 4$.) The truth table of this S-Box is given in Tab. 2.7.

The cipher is synthesized using various synthesis constraints into a 130 nm low leakage technology. The various logical netlists are translated into SPICE [78] netlists using extracted standard cells in BSIM3V3 model.

The cipher is fed with all the transitions of plaintexts and the currents $I(\text{VDD})$ and $I(\text{VSS})$ are extracted during the simulation with `elido` tool.

The exhaustive stimuli space exploration (2^{2q} traces) as well as the accuracy provided by the electrical simulation ensure that the traces we measure and the differential traces we compute emulate a perfectly noise-free DPA attack.

Table 2.7: The substitution box #0 $\{0,1\}^4 \rightarrow \{0,1\}^4$ of the encryption algorithm Serpent.

Inputs	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
Outputs	0x3	0x8	0xf	0x1	0xa	0x6	0x5	0xb	0xe	0xd	0x4	0x2	0x7	0x0	0x9	0xc

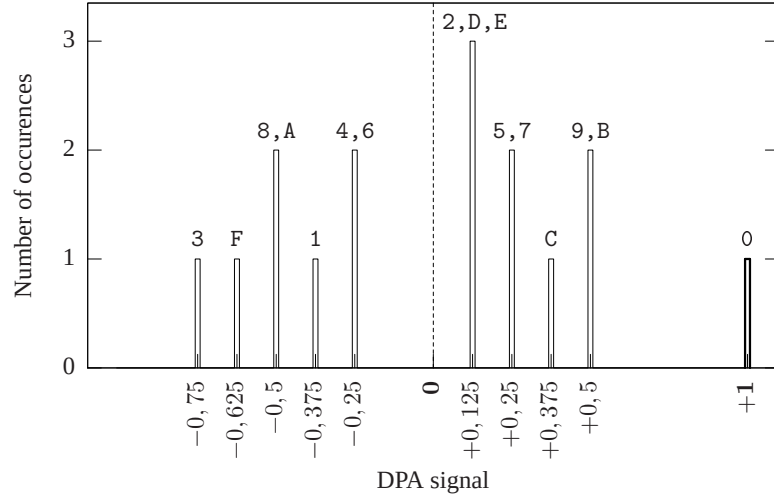


Figure 2.15: Theoretical histogram for the DPA signal (Eqn. 2.5). The hexadecimal values 0, $\dots$, F on top of the bars are those of $k_0 \oplus k$ (written on $p = 4$ bits). The thick peak for $k_0 \oplus k = 0$ is the DPA peak that betrays the secret key k_0 , whereas the others are the “ghost peaks”.

For the cipher described above, the theoretical model (Eqn. 2.5) predicts a DPA signal whose amplitude is given as an histogram in Fig. 2.15.

The differential traces depicted on Fig. 2.16 are computed from the traces acquired during the electrical simulation with the method explained above. The differential traces amplitude for the correct key can reach about 10 mA, which is also more or less the peak amplitude of a typical trace. The differential traces show that the amplitudes of secondary peaks are

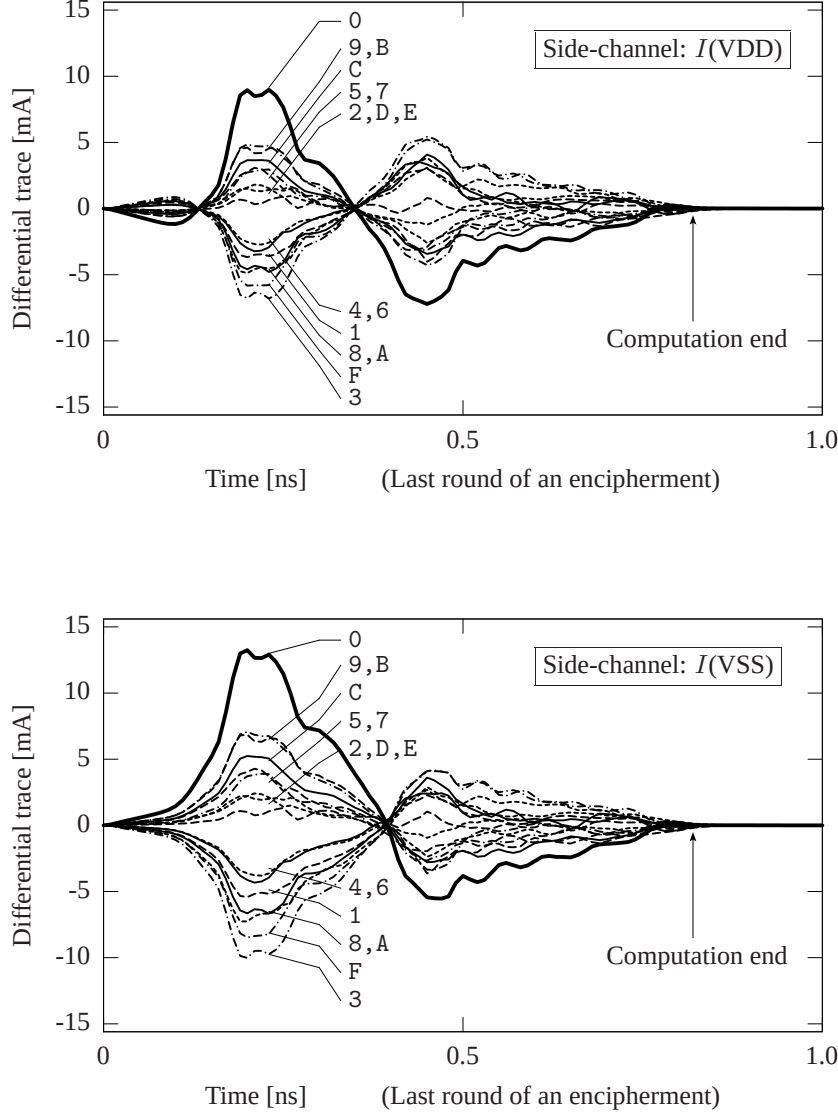


Figure 2.16: Electrical simulation of a DPA, using either $I(VDD)$ or $I(VSS)$ as the side-channel. The **bold curve** is the one computed when the hypothesis on the key is correct ($k_0 \oplus k = 0$).

those predicted by the histogram of Fig. 2.15 for both side-channel $I(VDD)$ and $I(VSS)$. This conclusion is the same for all the netlists we simulate, which tends to show that DPA does not depend on the implementation.

The DPA signals obtained by simulation match the theory, which justifies the DPA model

of Sec. 2.3.3 and proves that the difference $I_{\uparrow} - I_{\downarrow}$ of power consumption of a single gate can be extracted from the overall power consumed by a cryptographic operator. In addition, the model remains valid during much of the cipher computation time.

2.3.5 Connections between DPA and conventional cryptanalysis

2.3.5.1 DPA signal-to-noise ratio

DPA work factor is related to actual experiments, where the performance is assessed by a signal-to-noise ratio (SNR). As already mentioned, even if the DPA is not noisy, it does not allow to directly spot the right peak ($k = 0$) because there exists secondary peaks even for wrong keys ($k \neq 0$). Secondary peaks are modeled as noise. DPA quality is thus assessed by the following notion of SNR.

Definition 1 *Signal Sig SNR.*

$$\text{SNR}(\text{Sig}) \doteq \frac{\text{Sig}(k=0) - \overline{\text{Sig}}}{\left(\frac{1}{\#k} \sum_k (\text{Sig}(k) - \overline{\text{Sig}})^2 \right)^{1/2}}, \text{ where } \overline{\text{Sig}} \text{ is the signal mean.} \quad (2.7)$$

As far as the DPA signal (Eqn. 2.5) is concerned, a balanced S-Box F satisfies:

$$\begin{cases} \text{DPA}(0) &= \sum_{i,j} 2^{-p} \sum_x (-1)^{\langle \mathbb{1}_i \oplus \mathbb{1}_j | F(x) \rangle} \\ &= \sum_{i,j} \delta(\mathbb{1}_i \oplus \mathbb{1}_j) = q, \\ \overline{\text{DPA}} &= 2^{-2p} \sum_{i,j} \sum_x (-1)^{F(x)_i} \left(\sum_k (-1)^{F(x \oplus k)_j} \right) \\ &= 0 \quad (\text{because } F \text{ is balanced}). \end{cases} \quad (2.8)$$

As a result, the DPA signal-to-noise ratio is:

$$\begin{aligned} \text{SNR}(\text{DPA})(F) &\doteq \frac{\text{DPA}(k=0) - \overline{\text{DPA}}}{\sqrt{\frac{1}{\#k} \sum_k (\text{DPA}(k) - \overline{\text{DPA}})^2}} \text{ where } \overline{\text{DPA}} = \frac{1}{\#k} \sum_k \text{DPA}(k) = 0. \\ &= \frac{q}{\left(\frac{1}{2^p} \sum_k \left(\frac{1}{2^p} \left(\sum_{i=0}^{q-1} (-1)^{F_i} \otimes \sum_{j=0}^{q-1} (-1)^{F_j} \right) (k) \right)^2 \right)^{1/2}} \\ &= q 2^{\frac{3}{2}p} \left(\frac{1}{2^p} \sum_k \left(\widehat{\left(\sum_i (-1)^{F_i} \otimes \sum_j (-1)^{F_j} \right)}(k) \right)^2 \right)^{-1/2} \quad (\text{Parseval}) \\ &= q 2^{2p} \left(\sum_k \left(\widehat{\sum_i (-1)^{F_i}}(k) \right)^4 \right)^{-1/2} \quad \text{since } \widehat{f} \cdot \widehat{g} = \widehat{(f \otimes g)} \\ &= \boxed{q 2^{2p} \left(\sum_k \left(\sum_{i=0}^{q-1} \widehat{\theta}_F(k, \mathbb{1}_i) \right)^4 \right)^{-1/2}} \quad \text{since } \widehat{\theta}_F(k, a) = \widehat{(-1)^{\langle a | F \rangle}}(k), \quad (2.9) \end{aligned}$$

where:

- $\widehat{f}(k) \doteq \sum_x (-1)^{\langle x | k \rangle} f(x)$ is the “Fourier transform” of the Boolean function f (note that some authors use the same notation for the so-called “Hadamard-Walsh transform” of f , defined as the Fourier transform of $(-1)^f$) and

- $\theta_F(k, a)$ is the characteristic function of F , defined as $\delta(F(k) \oplus a) \doteq \begin{cases} 1 & \text{if } F(k) = a, \\ 0 & \text{otherwise.} \end{cases}$

The DPA SNR expression of Eqn. 2.9, proper to each S-Box F , fully characterizes the DPA discrimination power. Incidentally, it happens that the value of $\text{SNR}(\text{DPA})(F)$ (Eqn. 2.9) is significantly lower than $\text{SNR}\left(\sum_{i=0}^{q-1} \text{Corr}(k; \mathbb{1}_i, \mathbb{1}_i)\right)$ (refer to Eqn 2.3); for instance, those SNR are respectively 9.6 and 15.1 for AES. This proves that it is not realistic to neglect the inter-bit correlations (N3) when doing DPA. The available information in the traces is the sum of the consumptions of the q output bits of the function F and the DPA indeed only reveals the correlation of one bit of the predicted plaintext with the Hamming weight of the full plaintext.

2.3.5.1.1 SNR for some typical S-Boxes. A relevant reference for the SNR is the experimental setup where there is no S-Box. Analysis is thus performed behind a set of $q = p$ independent XOR gates. In this case, $\text{DPA}(k) = \sum_i (-1)^{k_i}$ (see Eqn. (2.10), with $F = I$, the identity matrix) and the SNR is $\sqrt{q}$.

The SNR of the DPA signal is computed using Eqn. (2.9) for balanced S-Boxes and an *ad hoc* calculus for the bent S-Box. Results are reported in Table 2.8. The SNR figures are given without any unit, because they are ratios of two commensurable quantities.

Table 2.8: Signal-to-noise ratio of DPA signal on some typical S-Boxes.

S-Box	No S-Box ($F=I$)	Linear S-Box	DES S-Box 1	AES	Bent S-Box
Fig. 2.17		(a)	(b)	(c)	(d)
p	8	8	6	8	8
q	8	8	4	8	4
DPA SNR	$\sqrt{8} = 2.8$	2.8	3.6	9.6	9.8

2.3.5.1.2 DPA SNR for an Affine Balanced S-Box. Let F be an affine balanced S-Box: $F(x) = M \times x \oplus D$.

Lemma 2

$$\text{Corr}(k; \mathbb{1}_i, \mathbb{1}_j) = (-1)^{\langle \mathbb{1}_j | M \times k \rangle} \delta_{i,j}, \quad (2.10)$$

thus: $\text{SNR}(\text{DPA})(F) = q^{\frac{1}{2}}$.

2.3.5.1.3 DPA SNR for a Bent S-Box. As far as (unbalanced) bent S-Boxes are concerned, DPA expression (Eqn. 2.5) yields high SNR (cf. Table 2.8). However, we have not investigated other expressions that could take advantage of the unbalancedness.

Results of Eqn. 2.8 do not apply to an unbalanced S-Box. Instead, if F is bent,

$$\left\{ \begin{array}{l} \text{DPA}(0) = q + \frac{1}{2^p} \sum_{i \neq j} (-1)^{\langle \mathbb{1}_i \oplus \mathbb{1}_j | F \rangle} (0), \quad \text{hence:} \\ |\text{DPA}(0) - q| \leq q(q-1)2^{-p/2} \ll q \quad \text{and} \\ \overline{\text{DPA}} = q2^{-p} \ll \text{DPA}(0) \quad \text{if } p, q \rightarrow \infty \text{ and } p \geq 2q \text{ [34].} \end{array} \right.$$

Therefore, at first order in $2^{-p/2}$, the expression of Eqn. 2.9 for DPA SNR still holds. It allows for the derivation of a minoration of $\text{SNR}(\text{DPA})(F)$:

$$\text{SNR}(\text{DPA})(F) \gtrsim q 2^{2p} \left(\sum_k \left(\sum_{i=0}^{q-1} 2^{\frac{p}{2}} \right)^4 \right)^{-1/2} = 2^{\frac{p}{2}}/q.$$

2.3.5.1.4 DPA SNR Bounds. Let us denote $Y_k = \left(\sum_{i=0}^{q-1} \widehat{(-1)^{F_i}}(k) \right)^2$.

$\text{SNR}(\text{DPA})(F)$ is thus rewritten as $q 2^{2p} (\sum_k Y_k)^{-1/2}$. The maximum and the minimum of the SNR correspond to the minimum and the maximum of $\sum_k Y_k$. Moreover,

Lemma 3

- If F is balanced: $\sum_k Y_k = q 2^{2p}$,
- otherwise: $\sum_k Y_k \in [0, q^2 2^{2p}]$.

• **DPA SNR maximum bound.**

The application $C : (x_0, x_1, \dots, x_{2^p-1}) \in (\mathbb{R}^+)^{2^p} \rightarrow \sum_{k=0}^{2^p-1} x_k^2 \in \mathbb{R}^+$ is convex. Its minimum is thus reached when C gradient is null, *i.e.* when all the x_k , $k \in [0, 2^p[$, are equal. Given Lem. 3, this value is $q 2^{2p}$ if F is balanced and can be as low as 0 otherwise.

Therefore, the maximum SNR of the DPA signal in a balanced S-Box is $2^{p/2}$. We ignore whether there exists S-Boxes that reach this bound. We also ignore whether DPA SNR is maximum bounded if the analyzed S-Box is unbalanced.

• **DPA SNR minimum bound.**

As $\sum_k Y_k^2 = (\sum_k Y_k)^2 - \sum_{k', k''} Y_{k'} Y_{k''}$, DPA SNR is minimum when the sum of positive terms $\sum_{k', k''} Y_{k'} Y_{k''}$ is minimum. It is null (and thus minimum) iff there exists an index k_0 such as all the Y_k , $k \neq k_0$, are null. If F is balanced, $\forall k$, $Y_k = q 2^{2p} \delta(k \oplus k_0)$. This lower bound can only be reached provided $q 2^{2p}$ (hence q) is a perfect square. If F is unbalanced, Y_k can reach $q^2 2^{2p}$. As for all $i \in [0, q[$ and $k \in K^p$, $\widehat{(-1)^{F_i}}(k) \leq 2^p$, Y_k reaches $q^2 2^{2p}$ iff for all i and k , $\widehat{(-1)^{F_i}}(k) = 2^p \delta(k \oplus k_0)$. S-Boxes satisfying this constraint are affine S-Boxes whose linear part rank is 1. Their corresponding SNR is $1/q$.

2.3.5.1.5 Summary of DPA Range and Typical Values. The results on the SNR of the DPA measured signal obtained in the Sec. 2.3.5.1 are summarized in Tab. 2.9.

Table 2.9: DPA signal-to-noise ratio bounds and typical values for different S-Boxes F .

SNR	Bound or typical value / S-Box type
$1/q$	Lower bound for unbalanced S-Boxes, reached only by rank 1 affine S-Boxes
1	Lower bound for balanced S-Boxes. Can only be reached if q is a perfect square
$q^{1/2}$	SNR of rank q affine S-Boxes
$2^{p/2}/q$	Approximative (at first order in $2^{-p/2}$) lower bound for bent S-Boxes
$2^{p/2}$	Upper bound for balanced S-Boxes

2.3.5.2 Conventional cryptanalysis evaluators

Algorithmic attacks, like linear [59] or differential [16] cryptanalysis, are measured by a maximum singularity in distributions. For example [34],

$$\begin{cases} \Lambda_S \doteq \sup_{a \neq 0, k} \left| \# \{x / \langle a | x \rangle \oplus \langle k | S(x) \rangle = 0\} - \frac{2^p}{2} \right|, \\ \Delta_S \doteq \sup_{k \neq 0, a} \# \{x / S(x) \oplus S(x \oplus k) = a\}, \end{cases} \quad (2.11)$$

are two parameters that characterize the resistances of an S-Box S against linear and differential cryptanalysis respectively. The lower they are, the more difficult the corresponding attack. We recall that in Eqn. ((2.2)), $F = S^{-1}$ for AES and $F = S$ for DES.

2.3.5.3 Comparing DPA and conventional cryptanalysis estimators

The results of the previous section tend to show that the less linear a S-Box (and thus the higher its cryptographic quality), the higher the DPA SNR. The histograms of the occurrences of the SNR signal amplitudes are shown for some S-Boxes in the section 2.3.7.

On the other hand, linear S-Boxes, the poorest protection against cryptanalysis, are the most difficult to attack by DPA.

The SNR of the DPA signal (Eqn. 2.9) is related to the two quantities Λ_S and Δ_S (refer to Sec. 2.3.5.2) that characterize linear and differential cryptanalyses on S-Box F by:

$$\text{SNR(DPA)}(F) \geq \frac{2^{\frac{3p}{2}-2}}{q \Lambda_S^2} = \mathcal{O}\left(\frac{1}{\Lambda_S^2}\right), \quad (2.12)$$

$$\text{SNR(DPA)}(F) \geq \frac{2^p}{\Delta_S} = \mathcal{O}\left(\frac{1}{\Delta_S}\right). \quad (2.13)$$

The best shielded against linear or differential cryptanalysis (Λ_S or Δ_S low), the more vulnerable to DPA attack ($\text{SNR(DPA)}(S)$ high). The proofs are given in the next two paragraphs.

2.3.5.4 Detail of the connection of DPA SNR with linear cryptanalysis

$$\Lambda_S \doteq \sup_{a \neq 0, k} \left| \#\{x / \langle a|x \rangle \oplus \langle k|S(x) \rangle = 0\} - \frac{2^p}{2} \right| = \sup_{a \neq 0, k} \frac{1}{2} \hat{\theta}_S(k, a) \quad [34]$$

hence

$$\forall k \in K^p, \forall a \in K^q \setminus 0, \quad \hat{\theta}_S(k, a) \leq 2\Lambda_S$$

thus:

$$\forall k \in K^p, \forall i \in [0, q[, \quad \left(\sum_{i=0}^{q-1} \hat{\theta}_S(k, \mathbf{1}_i) \right)^4 \leq (2q \Lambda_S)^4,$$

and finally the proof of Eqn. (2.12):

$$\begin{aligned} \text{SNR(DPA)}(F) &\geq q 2^{2p} \left(\sum_k (2q \Lambda_S)^4 \right)^{-1/2} \\ &= q 2^{2p} \underbrace{2^{-\frac{p}{2}}}_{(\sum_k)^{-1/2}} (2q \Lambda_S)^{-2} \\ &= \frac{2^{\frac{3}{2}p-2}}{q \Lambda_S^2} \\ &= \mathcal{O}\left(\frac{1}{\Lambda_S^2}\right). \end{aligned}$$

2.3.5.5 Detail of the connection of DPA SNR with differential cryptanalysis

$$\Delta_S \doteq \sup_{k \neq 0, a} \#\{x / S(x) \oplus S(x \oplus k) = a\} = \sup_{k \neq 0, a} \theta_S \otimes \theta_S(k, a) \quad [34]$$

$$\begin{aligned} \left(\sum_i \hat{\theta}_S(k, \mathbf{1}_i) \right)^4 &= \left(\sum_i \widehat{\theta_S(k, \mathbf{1}_i)} \cdot \sum_j \widehat{\theta_S(k, \mathbf{1}_j)} \right)^2 \\ &= \left(\left(\sum_i \theta_S(\cdot, \mathbf{1}_i) \otimes \sum_j \theta_S(\cdot, \mathbf{1}_j) \right)(k) \right)^2 \quad \text{since } \widehat{f} \cdot \widehat{g} = \widehat{f \otimes g}. \end{aligned}$$

As $\sum_k \hat{f}^2(k) = 2^p \sum_k f^2(k)$ (Parseval),

$$\begin{aligned} \sum_k \left(\sum_i \hat{\theta}_S(k, \mathbb{1}_i) \right)^4 &= \sum_k \left(\left(\sum_i \theta_S(\cdot, \mathbb{1}_i) \otimes \widehat{\sum_j \theta_S(\cdot, \mathbb{1}_j)} \right)(k) \right)^2 \\ &= 2^p \sum_k \left(\left(\sum_i \theta_S(\cdot, \mathbb{1}_i) \otimes \sum_j \theta_S(\cdot, \mathbb{1}_j) \right)(k) \right)^2 \end{aligned}$$

Given that when $\boxed{k = 0}$:

$$\begin{aligned} & \left(\sum_i \theta_S(\cdot, \mathbb{1}_i) \otimes \sum_j \theta_S(\cdot, \mathbb{1}_j) \right)(0) \\ &= \sum_{k'} \sum_{i,j} \theta_S(k', \mathbb{1}_i) \cdot \theta_S(k', \mathbb{1}_j) \\ &= \sum_{k'} \sum_i \theta_S^2(k', \mathbb{1}_i) \quad \left| \begin{array}{l} \text{if } F(k') = \mathbb{1}_i, \\ F(k') \neq \mathbb{1}_j \text{ when } i \neq j. \end{array} \right. \\ &= \sum_{k'} \sum_i \theta_S(k', \mathbb{1}_i) \quad \text{Remark that } \theta_S \in \{0, 1\}; \text{ Hence } \theta_S^2 = \theta_S. \\ &= \sum_i \# \{k' / F(k') = \mathbb{1}_i\} \\ &= q 2^{p-q} \quad \text{if } F \text{ is balanced (sane assumption),} \end{aligned} \tag{2.14}$$

and given that when $\boxed{k \neq 0}$:

$$\begin{aligned} & \left(\sum_i \theta_S(\cdot, \mathbb{1}_i) \otimes \sum_j \theta_S(\cdot, \mathbb{1}_j) \right)(k) \\ &= \sum_i \sum_{k'} \sum_{a \in \{\mathbb{1}_i \oplus \mathbb{1}_j, j \in [0, q]\}} \theta_S(k', \mathbb{1}_i) \cdot \theta_S(k' \oplus k, \mathbb{1}_i \oplus a) \\ &\leq \sum_i \sum_{k'} \sum_{a \in K^q} \theta_S(k', \mathbb{1}_i) \cdot \theta_S(k' \oplus k, \mathbb{1}_i \oplus a) \\ &= \sum_i (\theta_S \otimes \theta_S)(k, \mathbb{1}_i) \\ &\leq q \sup_{k \neq 0, a} (\theta_S \otimes \theta_S)(k, a) \\ &= q \Delta_S. \end{aligned} \tag{2.15}$$

Finally, using (2.14) and (2.15):

$$\begin{aligned}
& 2^p \sum_k \left(\left(\sum_i \theta_S(\cdot, \mathbf{1}_i) \otimes \sum_j \theta_S(\cdot, \mathbf{1}_j) \right)(k) \right)^2 \\
& \leq 2^p \left\{ \left(q 2^{p-q} \right)^2 + \underbrace{2^p - 1}_{\sum_{k \neq 0}} (q \Delta_S)^2 \right\} \\
& \leq 2^p q^2 \{ \Delta_S^2 + (2^p - 1) \Delta_S^2 \} \quad \text{because } 2^{p-q} \leq \Delta_S \text{ [34]} \\
& = q^2 2^{2p} \Delta_S^2.
\end{aligned}$$

Hence the proof of Eqn. (2.13):

$$\begin{aligned}
\text{SNR}(\text{DPA})(F) & \geq q 2^{2p} (q^2 2^{2p} \Delta_S^2)^{-1/2} \\
& = \frac{2^p}{\Delta_S} \\
& = \boxed{\mathcal{O}\left(\frac{1}{\Delta_S}\right)}.
\end{aligned}$$

2.3.6 Conclusion of the power attack model

The overall power consumption of a circuit leaks the activity of every single gate. The DPA attack exploits this side-channel to retrieve one secret kept within the circuit. The signal that an attacker computes to perform a DPA can be modeled as the auto-correlation of the Hamming weight of a given temporary variable used in the cryptographic algorithm. This auto-correlation function is maximum when the attacker key guess is correct. We have validated this model against an electrical simulation of a block cipher. The SNR of the DPA signal increases when the resistance against linear or differential cryptanalysis increases. The SNR is bounded, the lower bound being reached by the poorest cryptographic S-Boxes, namely affine S-Boxes. High quality cryptographic S-Boxes (AES, bent S-Boxes) feature high SNR, close to the maximum bound. As a consequence, DPA is fostered on devices implementing a high cryptographic-wise quality private key algorithm. The results from the DPA model presented in this Sec. 2.3 can be reformulated by the aphorism: resistance against **logical** attacks and resistance against **physical** attacks are **antinomic**. Similar works, by Emmanuel Prouff [77] and Claude Carlet [24], elaborate on this topic.

The same law can be drawn from the observation that not only the non-linear (S) parts of an algorithm, but also the linear (P) ones, can induce severe vulnerabilities. Gilles Piret's differential fault attack against SPN structures (like AES and Khazad) [40] illustrates this fact. Briefly, the working factor of Piret's DFA is that there exists many solutions K (prospective sub-keys to extract) to the byte-oriented equation:

$$S^{-1}(C \oplus K) \oplus S^{-1}(C^* \oplus K) = E,$$

where (C, C^*) is a public couple of a correct and faulted ciphertexts bytes and E an assumed error on one byte at the input of the last diffusion layer **ShiftRows** $\circ$ **MixColumns**. As a rule of thumb, if this equation had few solutions, then the differential characteristic Δ_S (2.11) of the sbox S , *i.e.* **SubBytes**, would be poor. But if the diffusion layer is good and that the fault occurs in the penultimate round, then many equalities (typically $i \in [1, 4]$ or $i \in [1, 8]$):

$$S^{-1}(C_i \oplus K_i) \oplus S^{-1}(C_i^* \oplus K_i) = E_i$$

hold concomitantly for related E_i (the E_i make up a linearly dependent family of 2^8 elements.) This coincidence, that arises from the good linear property of the permutation, makes it possible to enhance the key search.

Special care is thus needed while designing cryptoprocessors. As no trade-off is possible as for resistance against cryptanalysis, specific counter-measures must be devised. A possible counter-measure is to use secured logic gates [104]. However, those gates leak information because of parasitic effects: algorithmic counter-measures can thus be an adequate solution. For instance, the combination of a high SNR followed by a low SNR (in terms of DPA SNR) cipher on the same chip could provide a protection against both DPA and conventional cryptanalysis. Masking [58] and the duplication [41] method are other counter-measures that require to re-design the ciphers.

2.3.7 Illustration of DPA signal-to-noise ratio on histograms

The figures of this section show the histograms of occurrence of a given DPA signal amplitude. The actual signal is the peak of amplitude q (4 or 8), whereas the other peaks make up the S-Box intrinsic noise. It clearly appears in Fig. 2.17(a) that a linear S-Box has a weak SNR (namely $\sqrt{q}$). Usual cryptosystems DES (Fig. 2.17(b)) and AES (Fig. 2.17(c)) have a better SNR. The SNR is still better for a bent S-Box of Maionara-McFarland type [90] (Fig. 2.17(d).)

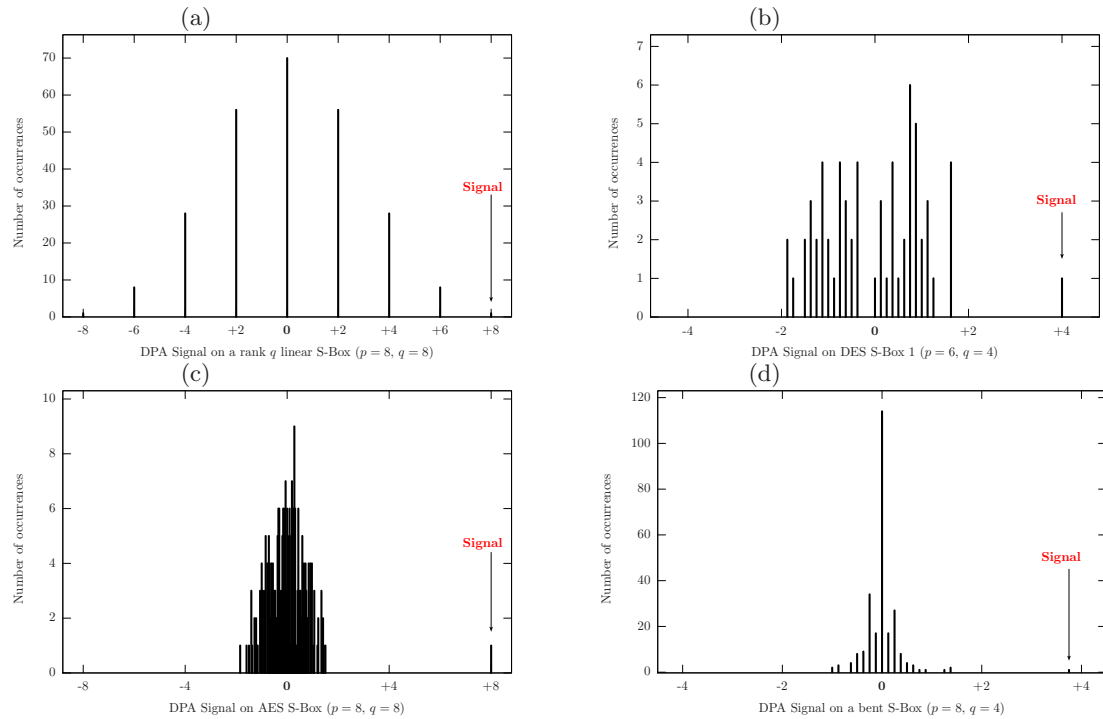


Figure 2.17: Histogram of occurrences of the DPA signal measured on:

- (a) a linear S-Box. $\text{SNR}(\text{DPA})(F) = \sqrt{8} \sim 2.8$ (Eqn. 2.9),
- (b) DES S-Box #1. $\text{SNR}(\text{DPA})(F) = 3.6$,
- (c) AES SubBytes. $\text{SNR}(\text{DPA})(F) = 9.6$,
- (d) a bent S-Box. $\text{SNR}(\text{DPA})(F) = 9.8$.

2.4 Practical computing as security weaknesses

In the previous section, we discussed how specific properties of cryptographic algorithms can enhance the side-channel attacks. In this section, the pragmatic aspects of computing are shown to make those attacks possible and to make them easier on embedded devices.

2.4.1 Integrability constraints

Louis Goubin, at page 162 of [41], and also Renaud Pacalet in the course [73] propose a methodology to analyze rationally the vulnerability against side-channel attacks. Their analytical approach is based on identifying “fundamental hypotheses” under which an attack is likely to be successful. Thanks to this methodology, we review here the usual integrability constraints that violate security “fundamental hypotheses”.

The realization of a SCA requires the following hypotheses:

H1 An exhaustive key search is possible.

H2 The attack does not require the knowledge of couples of plaintext and ciphertext.

Now, all cryptographic algorithms feature the following properties:

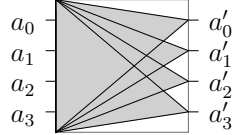
 The datapath is split into sub-paths, of typically $n = 6$ or 8 bits, because $\{0,1\}^n \rightarrow \{0,1\}^m$ functions, with $m \leq n$ are not integrable (and reasonably will not be integrable in a near future — unless an unprecedented technical breakthrough occurs.) An highly non-linear $\{0,1\}^n \rightarrow \{0,1\}^m$ vectorial Boolean function’s implementation roughly grows as $\mathcal{O}(m \times 2^n)$. This estimation holds for random functions, such as the DES sboxes. The implementation size might be reduced by using structured sboxes. For instance, the AES sbox’s area can be reduced by a factor four by taking advantage of its internal structure [82]. The statistics of Tab. 2.10 clearly show that synthesizers try hard to use as many cells as possible from the library for the straightforward look-up table (*a.k.a.* LuT) architecture, to the detriment of a global optimization (factored implementation.) The factored sbox is the number (1) in Fig. A.1(b), whereas the LuT is the number (2.) Still, from a purely mathematical point of view, it is difficult to find good primitives (security-wise and in terms of integrability.) There exists no general theory (out of existence theorems) to systematically construct highly non-linear mappings. For these reasons, the keys are split into n -bit sub-keys, which makes exhaustive searches on a 2^n space feasible.

 Given that the datapath divides into n -bit sub-paths, n bits of one inner round can be computed given 2^n hypothesis on a key. As the functions involved in one round have an appropriate non-linearity (refer to Sec. 2.3), the n -bit sub-key can be unambiguously extracted from only a one-round analysis. The keys used for every round are derived from the master key. However, to avoid reducing the key space, the generation of every round key is obtained from a stem that contains all the information about the key. Put differently, this means that every round key has the same quantity of information about the root key. As a consequence, the attack of the first or of the last round yields the same amount of information about the key. This information can be used to peel off another round. Alternatively, an exhaustive search on the remaining bits can be launched. However, the first solution is the more consistent, because all the necessary information has already been collected in the power traces.

Table 2.10: Area of the AES function `SubBytes` synthesized into a 717-element library from a 130 nm technology.

Implementation	# instances	# !instances	Area μm^2
Look-up-Table	423	53	4 018
Factorization in $\text{GF}(16^2)$	144	22	1 767

(a) Interpenetrating logic cones



(b) Internal logic cone to a net n_0

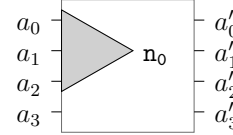


Figure 2.18: Illustration of the combinatorial logic factorization for a 4-bit $\rightarrow$ 4-bit mapping.

2.4.1.1 Attacks on hardware implementations

When the cryptographic algorithm can be implemented as a custom hardware, some constraints might be relaxed. For instance, an n to m -bit function can be optimized by sharing logic between the m Boolean functions. This interpenetration of combinatorial logic cones is illustrated in Fig. 2.18(a). In contrast, a ROM version cannot be shrunk below the nominal $m \times 2^n$ memory points. If larger functions are thus conceivable using combinatorial logical synthesis, an attack in the middle of the logic cone is thus becoming a new threat. With such an attack, a net that depends on a subset of the n inputs is analyzed, instead of an output that is a function of the n inputs. In Fig. 2.18(b), an internal net n_0 , depending on some but not on all the inputs, could be used to build a weighting function.

In pure hardware co-processors, there is also the possibility to choose the signals encoding. For instance, a dual-rail representation enables a key loading without dissipating its Hamming weight. As for the actual algorithm realization, constant Hamming weight functions should be used. However, the theoretical framework of this class of function does not exist yet. A trade-off is proposed in this thesis: using unbalanced functions, but with a dual-rail encoding, the overall function execution is balanced. This result is the subject of the chapter 4.

2.4.1.2 Attacks on software implementations

In software implementations of cryptographic algorithms, the vulnerabilities mainly arise from the underlying hardware that is in charge of executing the machine code. However, even for a properly compiled software, DPA is made possible on all the intermediate data due to an exhaustive search possibility (2^8 to 2^{32} is within reach of any attacker.)

The software is also intrinsically more vulnerable than dedicated co-processors because of its versatility. Thomas Messerges reports in [61] that the access to a memory location by the CPU of a smartcard signs by a voltage syndrome of 6.5 mV. In the DES cryptoprocessor analyzed in this dissertation, the signatures reach a few tens of millivolts only for the most loaded gates. For sure, between Messerges's experiments (1999) and ours (2005), six years, hence four technological generations [3], have elapsed. But this does not account for the three orders of magnitudes between versatile and dedicated power signature gap. For any operation, software implementations must active a lot of logic, even if it eventually selects only one value ; this is, in a nutshell, the reason why software implementations dissipate information with a

greater amplitude than tailored low-power hardware. Hardware-oriented architectures do not access a global RAM to fetch data (such as substitution boxes results), but instead work in local dedicated registers. Moreover, critical data is always computed locally (for instance sboxes are decomposed as combinatorial logic) in hardware, whereas static constant data are preferably lodged in a foreign RAM by software compilers.

2.4.2 Hamming weight *versus* Hamming distance selection functions

The previous section 2.3 concentrated on a DPA model based on the distinction of the signature of rising *versus* falling edges of gates transitions. The model can be adapted by changing the selection function (2.2).

Two distinct attack scenarios can be considered:

1. Only the value at a given date is known. This leads to the most powerful attack, because it requires the fewer information.
2. The previous and the current state of some variable are both known. This is a comfortable scenario, where the SNR of the attack is maximal. Attacks realized in this scenario are sometimes referred to as “CPA” [111, 112] instead of “DPA”. In the sequel, the generic acronym “DPA” is used to qualify both types of attacks.

In each case, the syndromes are:

1. $0 \rightarrow 0$ or $1 \rightarrow 0$ (*i.e.* $0 + I_{short}$, with the notations of Sec. 2.3.2.1) *versus* $0 \rightarrow 1$ or $1 \rightarrow 1$ (*i.e.* $0 + I_{short} + I_{load}$), in the case the spying resistor R_P monitors only the current delivered by the power supply.
2. $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$ and $1 \rightarrow 1$ are all known. The attack is less general but yields the correct keys with less power traces.

The signatures of the two transitions: $0 \rightarrow 1$ (rising edge) and $1 \rightarrow 0$ (falling edge) are evaluated with the following 32-bit selection functions:

1. $\overline{R_0} \cdot R_1$ (*rising edge of DES register R*) and
2. $R_0 \cdot \overline{R_1}$ (*falling edge of DES register R.*)

The two differential waves are represented in Fig. 2.19. It appears that it is very hard to distinguish a rising edge from a falling edge of a register. The highest difference occurs in clock cycle 18, where the rising (*resp.* falling) edge culminates at 1.39 mV (*resp.* 1.65 mV.) The optimal exploited bias are therefore:

- $|1.65 - 1.39| = 0.26$ mV for the Hamming weight attack and
- $|\frac{1}{2} \times (1.65 + 1.39) - 0| = 1.52$ mV for the Hamming distance attack².

Thus, in the best case, the syndrome is six times greater for the Hamming distance than for the Hamming weight selection function.

Hamming weight model applies well on combinatorial (or asynchronous) parts. Synchronous parts are vulnerable to the Hamming distance prevision attack, because it is well suited to multi-bit attacks (all the bits are valid simultaneously due to the synchronization by the global clock.)

Due to area optimization, most hardware implementations are devised to be iterative: this architectural strategy allows to re-use a common hardware resource as many times as possible.

²The distance from the average of the rising and falling value to zero is taken because $R_0 \oplus R_1 \doteq \overline{R_0} \cdot R_1 + R_0 \cdot \overline{R_1}$ and because the $0 \rightarrow 0$ and $1 \rightarrow 1$ transitions are assumed to dissipate no energy.

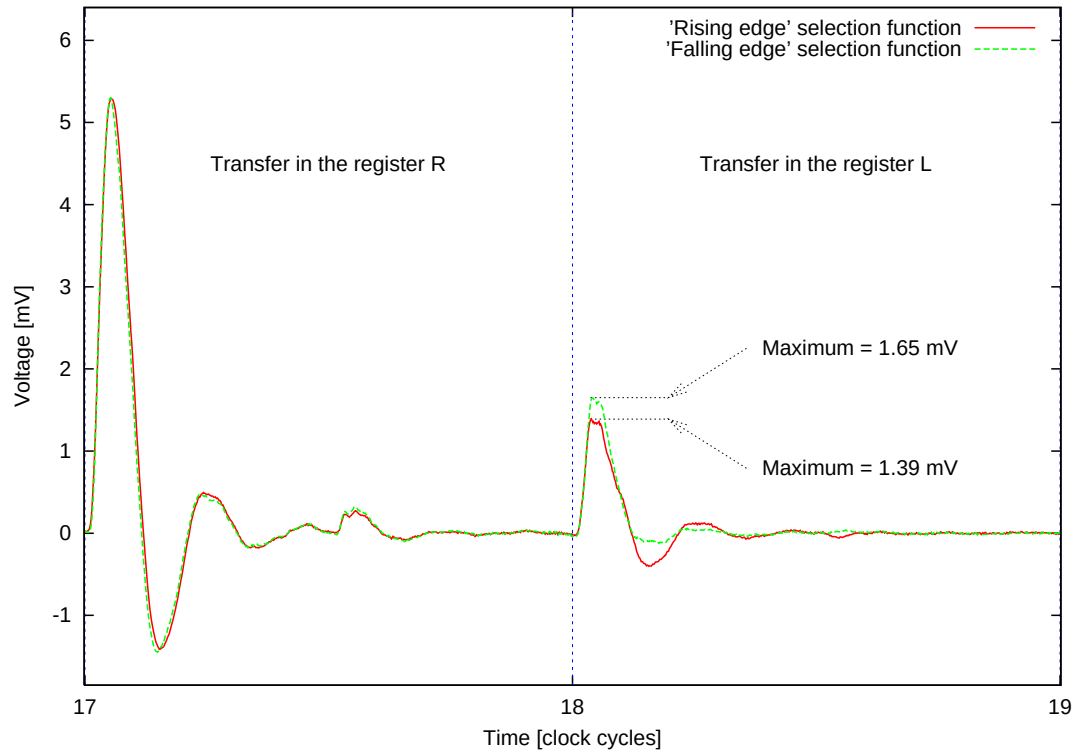


Figure 2.19: Differential traces resulting from the weighting by the two selection functions $|\overline{R_0} \cdot R_1|$ (*rising edge*) and $|R_0 \cdot \overline{R_1}|$ (*falling edge*) on DES.

With the notable exception of the combinatorial DES architecture exposed in Sec. 3.3, most hardware implementation of block cyphers execute in more than one clock cycle. This enables Hamming distance attacks.

From an acquisition campaign performed on an iterative DES hardware implementation, a collection of roughly 80 000 traces have been garnered. One typical trace is shown in Fig. 2.20. The Hamming weight and distance analysis have been performed on this set of traces, as shown in Fig. 2.21. The weighting functions are:

- $|\text{Sbox_}\#1(m, k)|$ for the Hamming weight (HW) weighting function and
- $|\text{R}_0\{9, 17, 23, 31\} \oplus \text{R}_1\{9, 17, 23, 31\}|$ for the Hamming distance (HD) weighting function.

The results for the seven other sboxes are given in appendix B.1, at page 162. The expected peak (indicating the looked for correlation) is marked by a vertical dotted black line on both graphs. The relevant date for the HW weighting function is the end of the sbx logic evaluation, that occurs about 8.45 ns after the rising edge of the first clock of the encryption (that corresponds to the sixteenth clock of the whole encryption process, as further explained in Sec. 3.2.) The HD weighting function reveals a peak when register R is overwritten by the first round result, which occurs at the very beginning of the clock period. Concretely, due to the modulation of the acquired signal by the probing system (the resonance frequency is about 150 MHz), the peak is slightly delayed: it takes its maximum 1.7 ns after the rising edge of the clock.

The other peaks are explained in the next section 3.6.

It is remarkable that:

- The HW peaks are less sharp than HD ones. This is due to the fact that the HD peak is synchronized by the clock: not only the four D-flip-flops $\text{R}_1\{9, 17, 23, 31\}$ sample at the same time, but they also do it for every trace. By contrast, the four bits output by the first sbx toggle at various times depending on the inputs and of the previous state of the sbx logic, but in addition the toggle time vary from trace to trace. This inevitably contributes to add random noise to the traces accumulation, thus reducing the peak contrast.
- In the best case (perfect synchronization of the traces), the SNR of the DPA signal is twice as low for the HW than for the HD:

$$\text{SNR}(\text{DPA}_{\text{HW}}) = \frac{1}{2} \text{SNR}(\text{DPA}_{\text{HD}}), \quad (2.16)$$

The result is that, in the HW case, only a *final* state is known, the *initial* state being unknown, whereas in the HD case, both *initial* and *final* states are known (given an hypothesis on the key.) The DPA_{HW} signal can be thought of as a DPA_{HD} signal averaged over all the initial states. However, in the practical cases, $\text{SNR}(\text{DPA}_{\text{HW}}) \ll \frac{1}{2} \text{SNR}(\text{DPA}_{\text{HD}})$, because of the temporal dispersions of the (power consuming) monitored events. Incidentally, a physical argument deserve consideration: in HW, the leakage has a “technological” origin (refer to Fig. 2.11), whereas in HD, the leak is of “logical” nature: at the first order, a DFF does not dissipate any energy when its state is not changed, while it consumes power when toggling state. Eventually, it could be thought that HW types of attacks leave more room for creativity, since any function $f(m, k)$ can be used, whereas f is fixed to $f_{\text{HD}} \doteq \text{DFF}_i \oplus \text{DFF}_{i+1}$ for some extremal round index i when using HD. However, as it will be clear in Sec. 3.7, the choice f_{HD} is indeed really good, at least in the straightforward iterative implementations. Of course, an attacker can customize its distance computation, but the simple f_{HD} weighting function already enables an excellent discrimination power between the correct and the wrong key guesses.

A proper measurement of $\text{SNR}(\text{DPA}_{\text{HW}})$ is not possible 8.45 ns inside the first round, because the correct key does not yield the highest differential trace. If we resort to the peak

in the next clock period, we measure $\text{SNR}(\text{DPA}_{\text{HW}}) = 3.95$. In the same clock period, we also measure $\text{SNR}(\text{DPA}_{\text{HD}}) = 8.11$, which validates the Eqn. (2.16).

Finally, it is also worth noticing that correlations in iterative designs might be exploitable over two clock periods. This issue is seldom discussed in the available literature. The up-coming publication [75] is concerned with this aspect in EMA. We demonstrate the same result on DPA, on an iterative DES straightforward hardware implementation. The three graphs of Fig. 2.22 show differential traces (for the correct key $k_0 = 0\text{x}56$), using the following weighting functions:

1. $\text{LR}_0(m, k) \oplus \text{LR}_1(m, k)$,
2. $\text{LR}_0(m, k) \oplus \text{LR}_2(m, k)$,
3. $\text{LR}_0(m, k) \oplus \text{LR}_3(m, k)$.

The most significant information can be extracted from the traces weighted by the regular HD selection function (Fig. 2.22(1)). The amplitude of the main peak is 10.56 mV. This figure corresponds to eight times the amplitude of the peak of the *lower graph* of Fig. 2.21 (DES has eight substitution boxes), that is to say $8 \times 1.18 = 9.47$ mV, plus the much less power dissipating (in because the fan-out is only one) direct 32-bit register transfer $L_0 \rightarrow L_1$.

The maxima for the eight sboxes (S1 to S8) and for the transfer $L_0 \rightarrow L_1$ can be computed:

Register transfer's HD	S1	S2	S3	S4	S5	S6	S7	S8	$L_0 \rightarrow L_1$
Maximum [mV]	1.18	0.79	0.68	0.88	0.94	1.18	0.91	1.17	3.01

The sum of these figures is equal to 10.73 mV, *i.e.* slightly more than the measured 10.56 mV, because all the peaks do not occur exactly at the same date. The discrepancy expresses a triangular inequality.

If, instead, the integral of the differential traces over the 17th clock period are computed, then there is a perfect equality.

Register transfer's HD	S1	S2	S3	S4	S5	S6	S7	S8	$L_0 \rightarrow L_1$
Integral [mV $\times$ ns]	1.79	0.62	0.92	1.12	1.67	1.96	1.19	2.06	4.95

The integral is equal to 16.28 mV $\times$ ns when summing together the figures from the previous table. This is also the result of the integration over $t \in [\text{clock}_{17}, \text{clock}_{18}[$ of the differential trace from Fig. 2.22(1).

However, Fig. 2.22(2) shows that the distance-2 Hamming difference still leaks out some information. There are two ways to interpret this observation:

1. either the DFFs somehow memorize some of their states even after one erasure
2. or an event that signs as $\text{LR}_0 \oplus \text{LR}_2$ occurs in the datapath.

As argued in Sec. 3.6.1, the principal reason is a peculiarity of DES, and not of a parasitic memorization.

Higher distances, starting from 3 (Fig. 2.22(3)), do not reveal any further information.

In the rest of this thesis, and unless otherwise mentioned, the weighting function used for the analysis will be the *unitary* Hamming distance.

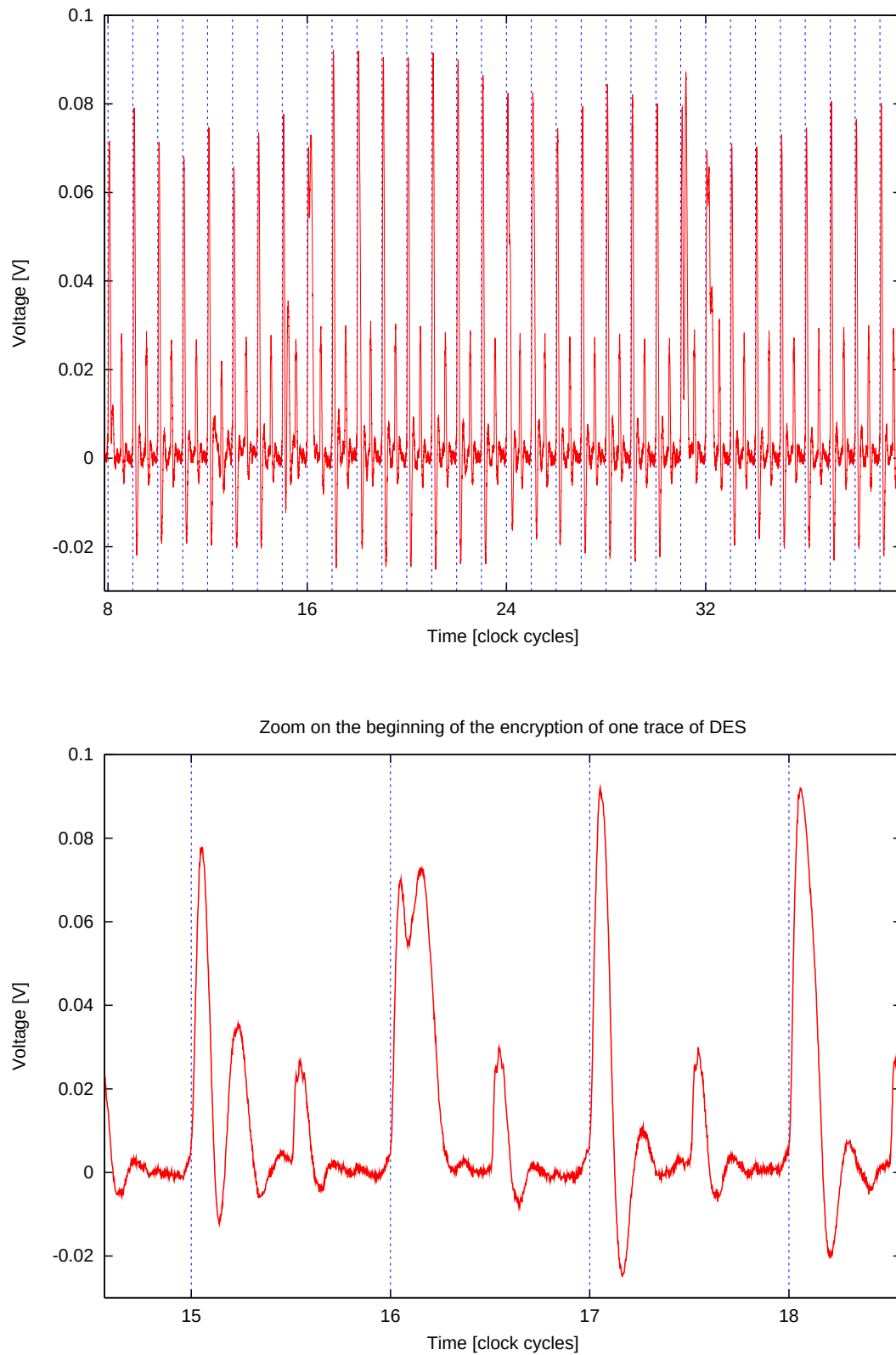


Figure 2.20: Typical power trace used to perform Hamming weight *versus* Hamming distance functions comparison. The *upper* graph shows the power signature of a full DES encryption (the sixteen rounds unroll between clocks 16 and 32), while the *lower* graph focuses on the first round under attack (thus corresponding to the clock interval 16–17.)

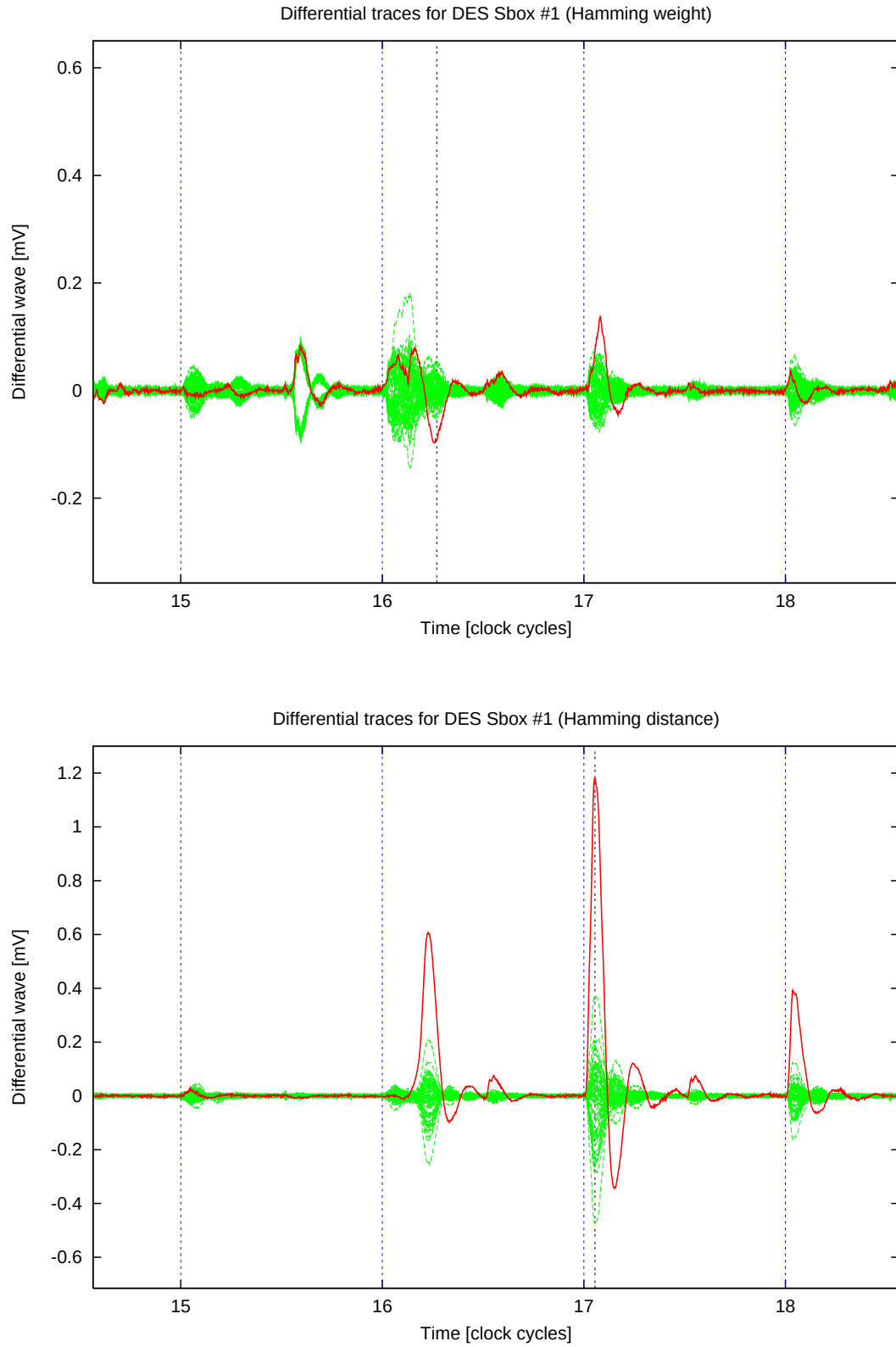
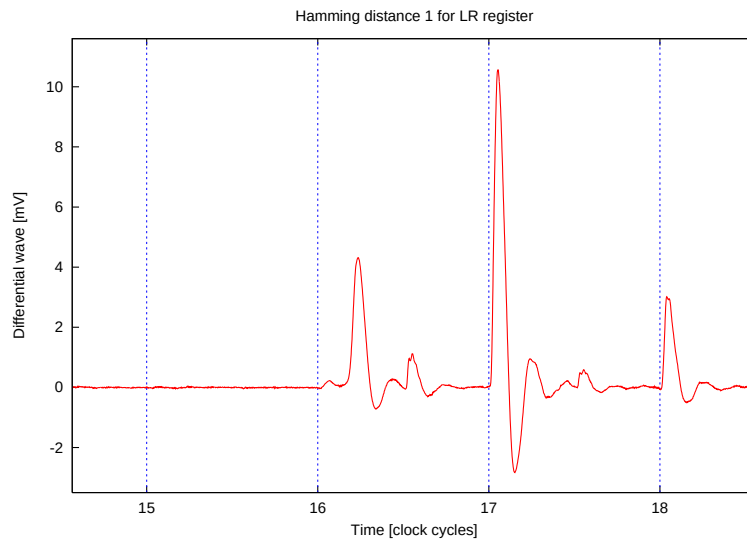
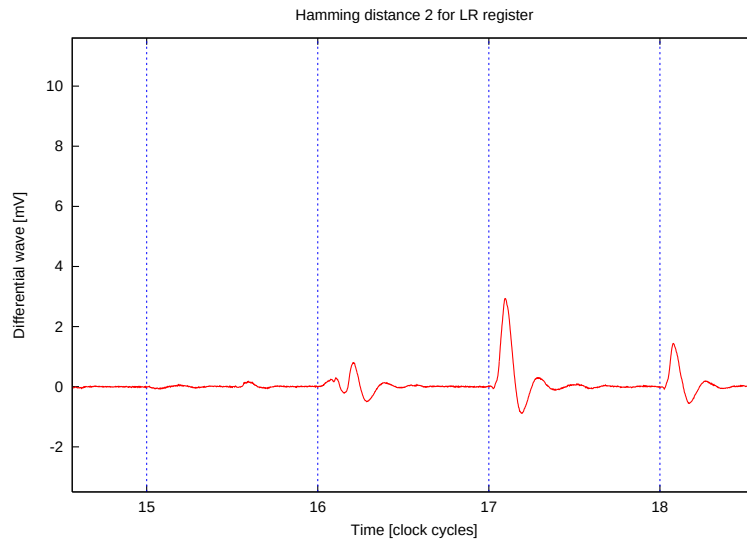


Figure 2.21: Comparison between an Hamming weight (*upper graph*) and an Hamming distance (*lower graph*) weighting functions. The correct key ($k_0 = 0x56$) is drawn in **bold red**.

(1)



(2)



(3)

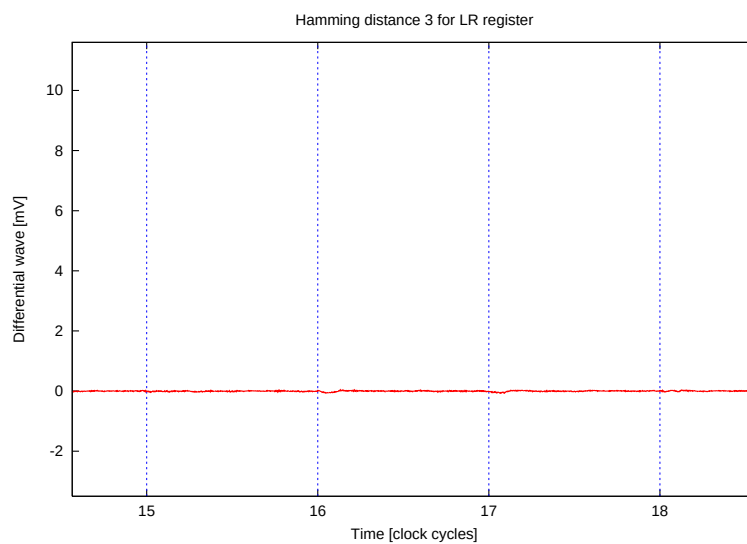


Figure 2.22: Hamming distance 1, 2 and 3 from the first round on the whole 64-bit register LR of DES.

Chapter 3

DPA on DES

The previous chapter showed that whenever some physical information can be extracted from a device, the security of the latter is jeopardized. This chapter aims at characterizing the risks: the factors that make attacks possible are brought to the fore. First of all, a full-featured architecture of a DES crypto-processor (able to perform simple and triple encryptions in conjunction with all the standardized modes of operations) is described. Second, some known logical weaknesses of the DES based on weak and semi-weak keys are explicated. Then, the logical weakness is exploited by a side-channel analysis using only two power traces to characterize the information leaks. The leaks reveal the algorithm's architecture, thus allowing to customize an attack. Finally, additional results from a real DPA are presented.

3.1 Secured crypto-processors design

3.1.1 Security target

When devising a “trusted computing platform”, the asset to protect can be an ASIC, an FPGA, a CPU, or even a “true” random number generator (T-RNG – analog device.) However, these devices are always a piece of hardware. The virtuous security construction mechanism consists in enlarging the trusted zone. This idea is common to:

- Processors working with untrusted hardware components, such as CryptoPage [55] or AEGIS [98].
- Pieces of hardware certified by a trusted module, such as TCG [6] (the former “Trusted Computing Platform Alliance”, *aka* TCPA.)
- And in general, any public-key cryptography, giving rise to PKIs.

This hypothesis is always that there exists a secured zone that is the stem to the global security. The “root of security” is a germ for any large scale “trusted computing platform”.

In the sequel, we focus on the securization of a DES co-processor. Its security features are studied in this chapter, and the means to increase its security is the subject of the next one.

3.1.2 Motivation for DES

DES is certainly the most studied symmetrical algorithm, and is currently widely used (in the form of triple-DES or DESX [84] in low-cost smartcards). It is forecast that it will probably still be around in ten or twenty years from now [26].

The algorithm uses eight different sboxes S1 to S8, which allows us to realize comparisons between their relative strength with the same set of traces.

DES has been designed to be very efficient in hardware (all the permutations and substitution boxes are indeed a nightmare to write on 8-bit machines), whereas AES is efficient in software (a 919 bytes implementation is described in [72]) From a hardware design point of view, DES is the best candidate, because it is small and versatile: one DES engine can be used to perform encryption, hash, MAC and random number generation.

The studied implementation is described in details in the following section.

3.2 A fast pipelined multi-mode DES architecture operating in IP representation — Integration, the VLSI Journal article extended version [101]

The Data Encryption Standard (DES) is a cipher that is still used in a broad range of applications, from smartcards, where it is often implemented as a tamper-resistant embedded co-processor, to PCs, where it is implemented in software, for instance to compute `crypt(3)` on UNIX platforms. Notice that `crypt` features a security enhancement over DES: it makes use of a 12-bit salt and of 25 consecutive encryptions. System utilities, like `passwd(1)` or `htpasswd(1)`, can be based over `crypt` or MD5 [83]. To the authors' knowledge, implementations of DES published so far are based on the straightforward application of the NIST standard. This section describes an innovative architecture that features a speed increase for both hardware and software implementations, compared to the state-of-the-art. For example, the proposed architecture, at constant size, is shown to be about twice as fast as the state-of-the-art for 3DES-CBC (refer to Fig. 3.10.) The first contribution of this section is an hardware architecture that minimizes the computation time overhead caused by key and message loading. The second contribution is an optimal chaining of computations, typically required when “operation modes” are used. The optimization is made possible by a novel computation paradigm, called “IP representation”.

3.2.1 Introduction on DES

The Data Encryption Standard, DES, is a block product cipher algorithm promoted by the NIST. The latest version of the standard is known as FIPS 46-3 [71], and includes the definition of “triple DES”. The “DES modes of operation”, standardized in FIPS 81 [67], is a companion document devoted to the description of the secure use of DES when the messages to encrypt are longer than 8 bytes.

DES is an modification of the algorithm Lucifer developed by IBM in the early 1970s. Since its inception, DES has been used pervasively by many applications that require data confidentiality. Since 2001, DES has been superseded by the Advanced Encryption Standard AES [72]. However, in practice, a lot of hardware or software applications still resort to DES.

The DES algorithm turns a 64-bit confidential data block, nicknamed *plaintext*, into another 64-bit data block, nicknamed *ciphertext*, using a standardized bijection parametrized by a 56-bit secret, nicknamed *key*. The bijection DES_k is crafted in such a way that it is almost impossible to retrieve the plaintext from the ciphertext without the knowledge of the key k . The bijection can be inverted: this operation is called *decipherment* and noted DES_k^{-1} . When it is not relevant whether the algorithm performs encipherment or decipherment, the neologism “*cipherment*” is employed instead.

Several attacks against the plain DES version were published. They can basically be classified into two categories: *algorithmic* and *physical* attacks.

3.2.1.1 Algorithmic attacks on DES

Algorithmic attacks are also referred to as cryptanalysis [16, 59]. Those analyzes are somehow unpractical, since a large amount of {plaintext, ciphertext} couples must be intercepted. The exhaustive search of the key has been *publicly* feasible since 1997, as proved by the RSA Laboratory’s “DES Challenge II” being won in 1997 in 39 days by a network of computers running the distributed application DESCHALL [27] and in 1998 in 3 days by a dedicated machine built by the EFF [28]. Other methods to speed-up the search using pre-computed datasets have been put forward [51].

To counteract those attacks, variants of the DES were proposed. We list below three of the most widespread ones:

1. **Modes of operation** allow a message consisting of several 64-bit blocks to be ciphered in chain. The idea is that the 64-bit ciphertext blocks actually depends on the corresponding plaintext block and also of some, if not all, of the previous ones, and of an initialization vector (IV.) The standardized modes of operation are ECB, CBC, CFB and OFB [67]. ECB and CBC are *block-ciphers*, whereas CFB and OFB are *stream-ciphers*. The latter two are actually defined in the K -bit version, $1 \leq K \leq 64$. As the $K = 64$ version is the most efficient in terms of throughput, it is usually the sole version to be implemented (refer for instance to `openssl` [30].) Because of the “short cycle property”, NIST explicitly does not support $K < 64$ for OFB [70, page 13].
2. **TDEA (informally called “triple-DES” or “3DES”)** is described in the annex of the DES standard [71, page 22]. Three 64-bit keys $k_i, i \in \{0, 1, 2\}$ are used instead of one. The encipherment consists in computing $\text{DES}_{k_2} \circ \text{DES}_{k_1}^{-1} \circ \text{DES}_{k_0}$ whereas decipherment is $\text{DES}_{k_0}^{-1} \circ \text{DES}_{k_1} \circ \text{DES}_{k_2}^{-1}$. Triple DES is customarily used with two keys [79] ($k_0 = k_2$.) Notice that when the three keys are taken equal, $k_0 = k_1 = k_2$, triple DES actually computes plain DES, which guarantees the backward compatibility of 3DES engines.
3. **DESX [52]** is a data whitening technique, proposed by Ron Rivest. It consists in adding two 64-bit blocks, `in_white` and `out_white`, to the key. The key `in_white` is used to *exclusive-or* (i.e. XOR) the plaintext prior to starting DES and `out_white` to XOR the result after the cipherment.

Those variants can of course be combined at will. For instance, triple-DES using two keys in CBC mode is often used to encipher long messages. Any hardware implementation of DES must nowadays support the modes of operation and the triple encryption.

3.2.1.2 Physical attacks on DES

Physical attacks are the most recent threats against DES and its variants. The side-channel attacks, such as DPA [54] or EMA [37], allow to retrieve the keys by the analysis of the physical emanation of the device while it is handling the key. Partial side-channel information, such as the Hamming weight of key chunks or key-dependent correlations between two small chunks of data, suffice to recover the full key, provided enough measurements can be performed. The faults injection attacks [17] consist in either perturbing transiently the circuit or to damage it to enhance other attacks. Algorithmic counter-measures (modes of operation, 3DES or DESX) do not protect against physical attacks. Both side-channel and fault attacks can be thwarted, with more or less success, by using leakage-proof logic and adequate sensors, for instance.

In this section, we describe an architecture able to compute DES and its variants efficiently. More precisely, the described architecture can compute: DES in ECB, CBC, 64-bit CFB and 64-bit OFB, as well with simple or triple DES using two keys. The cryptanalytic strength of the variant as well as the security of its implementation against physical attacks is out of the scope of this section.

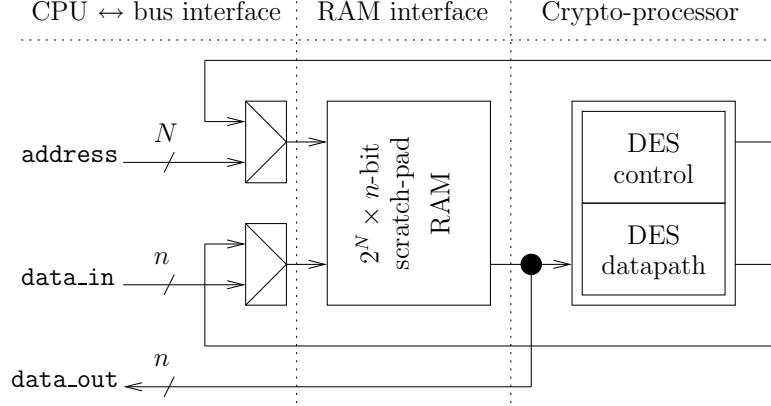


Figure 3.1: System-on-Chip environment for a VLSI version of the DES co-processor. Typical values for the bus widths are $n = 8$ and $N = 8$.

The rest of the section is organized as follows. Section 3.2.2 discusses the DES datapath optimization: an hardware pipelined architecture is presented. Section 3.2.3 applies to both software (SW) and hardware (HW) implementations. It introduces the so-called “IP representation” computational framework, which allows to optimally chain DES computations. In Sec. 3.2.4, the gain of proposed architecture over state-of-the-art architectures is discussed. Finally, Sec. 3.2.7 summarizes the architecture study.

3.2.2 DES datapath improvement thanks to a generalized pipelining

In the DES algorithm, the control is independent of the data. It is thus safe to consider the design of the datapath and the control finite state machine (FSM) as two distinct tasks. This section is devoted to the datapath. The control is further studied in Sec. 3.2.3.

3.2.2.1 Straightforward DES

The inputs of the DES algorithm are two 64-bit blocks, the plaintext and the key¹. The two operands cannot be loaded in the DES operator in one go, since data provided by processors are typically on $n = 8, 16$ or 32 bits. In the rest of the article, we assume that the DES co-processor is fed by an $n = 8$ -bit wide data bus. This figure corresponds to the case of an embedded system built around a micro-controller, as depicted in Fig. 3.1. This architecture integrates naturally into the SoC SecMat V1 as a module (refer to Tab. A.1 in appendix.)

Most of DES implementations elude the question of the connexion to an $n < 64$ wide bus [44, 92, 21]. Other implementations, such as [10], do not take advantage of the architectures presented in this study.

The knowledge of the DES algorithm internals is not required to explain the rationale of the three implementations discussed in this section. Only the following facts are indeed relevant for the coming analysis:

- DES is a Feistel cipher [45], which means that the message is divided into two halves (L and R), among which only L undergoes a logical operation dependent on the some bits of the round key, R being left untouched. Then the two halves are swapped, and the process is iterated sixteen times. After the last round, L and R are not swapped.

¹The entropy of the key is actually limited to only 56 bits, because 8 bits are redundant.

- Before any processing, the message bits are shuffled, using a permutation called IP. At the end of the Feistel scheme, the message is de-shuffled by the inverse permutation $FP \doteq IP^{-1}$.
- Only 56 bits of the key are used. As justified in the standard [71, page 1], every byte of the key has a parity bit, chosen so that the Hamming weight of every byte of the key is odd. In a similar way to the message, the key bits are initially shuffled, using the permutation PC_1 . A round key K_i is extracted from K at each round i ; it is stored in a register called CD. The K_i are transformed by an operation known as “key schedule”, consisting in one or two Left Shifts, LS (resp. Right Shifts, RS) for encipherments (resp. decipherments), followed by a permutation called PC_2 . The shifts are designed in such a way that CD is back to its initial value after a full encipherment. They are implemented by a 2×2 input multiplexor ($4 \rightarrow 1$ MUX.) However, when enciphering, the initial value to be presented at PC_2 is $LS(k)$, whereas when deciphering, bare $Id(k) \doteq k$ is to be used instead. Given that a “general purpose” DES module is designed to both encipher and decipher, both PC_1 and $LS \circ PC_1$ must be computed in parallel.

As a result, a straightforward implementation of DES requires the following sequential resources:

1. one 64-bit register (named LR in [71, page 11]) to hold the ciphertext and to store the 16 intermediate round outputs, and
2. one 56-bit register (named CD in [71, page 19]) to hold the key stripped off its parity bits and to store the 16 round keys.

Without any additional registers, the storage of the plaintext in LR and of the key in CD requires a demultiplexing logic, illustrated in Fig. 3.2. For the sake of clarity, the control part has been omitted in Fig. 3.2: the multiplexors and the key schedule logic are *implicitly* commanded externally.

The schematics follow those conventions:

- sequential gates, Flip-Flops (DFF) in our case, are represented as boxes (■),
- combinatorial gates are represented as boxes with round corners (□ or ■),
- permutation-only gates, such as IP or buses merge ($\curvearrowright$) or split ($\curvearrowleft$), are hollow, whereas
- gates made up of logic have a solid background ;
- datapath forks are represented with solder dots (●) and
- when some bits are useless, they are disposed (⊠).

The entire DES design is made up of bit shuffling dataflow primitives (permutations, multiplexors and flip-flops), with the exception of the round logic. This fact is depicted on Fig. 3.3, where the critical path of the datapath is highlighted. The typical resource utilization in the straightforward architecture of Fig. 3.2 is illustrated in Tab. 3.1.

Registers LR and CD must be loaded sequentially. In a pipelined architecture, the use of “enable” signals on the DFFs can usually be avoided. It is possible to use none, if the key is loaded first into CD, because there is a way to keep it “apparently” still. As $LS^2 = LS \circ LS$, $RS^2 = RS \circ RS$ and $LS \circ RS = RS \circ LS = Id$, it is easy to control the key in such a way it is unchanged before and after the LR loading. In the sequel, we assume that the transformation is $LS^4 \circ RS^4$.

As for LR, it never has to maintain its state more than one clock cycle. The same remark will hold for the refinements carried out on this straightforward architecture, because they are

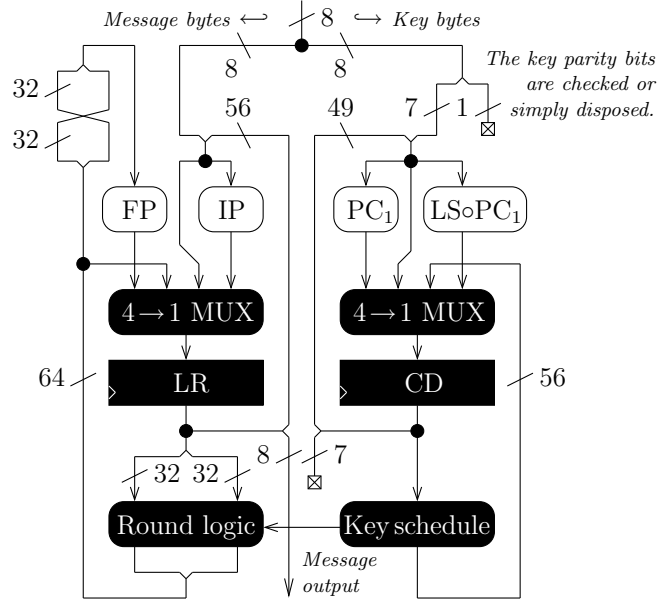


Figure 3.2: Straightforward architecture for a DES module, equipped with demultiplexing logic to load the message and the key one byte at the time.

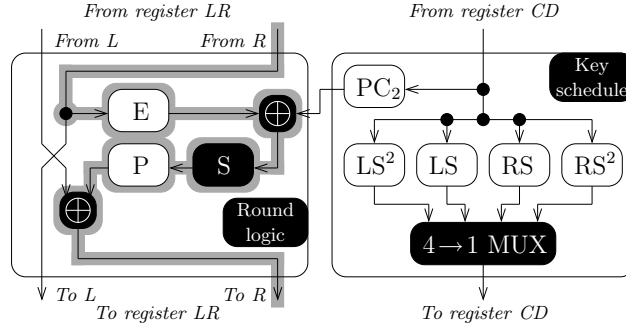


Figure 3.3: DES round and key schedule combinational logic. The critical path $LR \rightarrow \text{Round Logic} \rightarrow LR$ is highlighted.

Table 3.1: Resources area [μm^2] in Fig. 3.2, synthesized at 400 MHz in a 130 nm ASIC low-leakage technology.

Datapath				Control
Round logic	Rest: Dataflow logic			
S + XOR	Permutation	MUX	DFF	FSM
8482	0	7193	3437	5075

“pipelined”: data (other than the key) flows continuously through the datapath, without having to wait at any time.

The straightforward pipeline is thus initially busy during $64/n = 8$ clock cycles to load the key into CD. During another eight clock cycles, the key is applied $LS^4 \circ RS^4$, whilst the first message block is loaded into LR. Then the DES engine starts the sixteen iterations. The next eight clock cycles are devoted to flushing the result out.

In the straightforward scheme of Fig. 3.2, every computation has an overhead in execution time due to data loading / unloading in the LR or in the CD register. The evaluation of the architecture throughput does not take into account the key loading, because most applications use only one key, loaded once for many consecutive cipherments (the case of 3DES is detailed later in Sec. 3.2.3.2.) The loading stage consumes $64/n = 8$ cycles, and monopolizes the LR or the CD registers, so that it is impossible to parallelize a loading with a DES cipherment (16 cycles). Then the message must be output, which requires another $64/n = 8$ cycles. Notice that for read and write accesses to be done in parallel, two RAMs must be connected to the DES engine. In terms of memory usage, it is however optimal to use a single RAM, since every computation result can be written over the original message. Thus, the maximum throughput is one encipherment per $8+16+8$ clock cycles (2.0 bit/clock). With a double port RAM, the write and read accesses could be parallelized, which would reduce the encipherment time to 24 cycles.

The straightforward architecture suffers two drawbacks, that impede the cryptoprocessor performances:

1. The DES cannot perform cipherments while new blocks m_{i+1} are being read and processed blocks $DES(m_i)$ are being written out.
2. The LR register is preceded by two layers of multiplexors, that increase the critical path.

The next section describes and motivates a novel pipelining scheme, where the data can be transferred byte by byte, in parallel with DES cipherments.

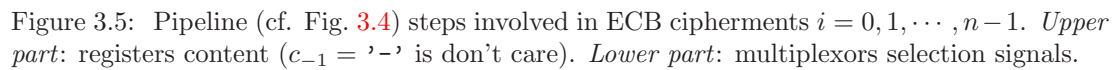
3.2.2.2 DES datapath fast pipelining

The drawbacks put forward in the previous section can be overcome by a more elaborate pipelining scheme of the DES cryptoprocessor. The principle is to parallelize the message loading and unloading with the DES algorithm itself. A comparison between the so-called *iterative* and *pipeline* architectures of DES inner-loop is discussed in [81, page 589]. The difference is that an iterative DES engine processes one cipherment at the time, whereas a pipeline DES engine can process many – up to 16 – at the same time. In all the architectures presented here, DES is computed iteratively. However, the outside view of the DES engine is more like a pipeline: data is not input and output monolithically, but rather byte by byte. It must be clear that, throughout this study, the term “pipeline” refers to the way the data is loaded and unloaded.

A 64-bit register, called IF (because of its role of InterFace between the 8-bit inputs and the 64-bit blocks involved within DES), is added to the DES cryptoprocessor.

IF is designed to have two possible sources: it can input either individual bytes or 64-bit blocks. In the first case, the output of IF is shifted by 8-bits to make room for the incoming byte, to be concatenated with the others already collected. The byte that has been “shifted-out” is not lost: it is available at the eight-bit output of the pipeline. In the second case, a 64-bit block, such as the result of the DES computation, is latched into IF, in a view to being output byte by byte. In the meantime, the whole content of IF can be transferred to LR, so that the DES datapath is ready to immediately start another cipherment.

The same IF register can be reused to manage the 8-bit $\leftrightarrow$ 64-bit conversion for both LR and CD. Figure 3.4 illustrates that the pipeline is generalized to cover both the round logic and the key schedule.



A more detailed description of the pipelined process is given below and illustrated in Fig. 3.5 for DES-ECB encipherment with one key:

- 1–7: During seven clock periods, the seven first bytes of the key k are loaded, side-by-side, into IF.
- 8: The blocks comprised of the last byte of the key $k[56, 63]$, concatenated with the already loaded seven others $k[0, 7] || \dots || k[48, 55]$, are then loaded into CD, using selection 0 (when deciphering) or selection 1 (when enciphering).
- 9–15: During the seven following clock periods, the message m_0 is built-up into IF.
- 16: The message m_0 , now complete, is transferred into LR. In the meantime, k is kept still in CD, which is possible, as shown in Sec. 3.2.2.1. At the same time, the result $\text{DES}_k(m_{-1})$ of the previous computation – if any – is latched into IF.
- 17–24: The next eight cycles would be devoted to the output of an hypothetical $c_{-1} \doteq \text{DES}_k(m_{-1})$, byte by byte ($c_{-1}[8 \cdot i, 8 \cdot (i + 1)[$, $i \in [0, 8[$), from IF. In the present case, c_{-1} is a “don’t care” result. However, starting from clock cycle 33, relevant c_i , $i \geq 0$ will be delivered byte by byte from IF. Concomitantly, the first eight rounds of DES are executed.
- 25–31: Whilst DES rounds are computed, a new 64-bit block of data is loaded (as already seen at clock cycles 9–15.)
- 32: DES has finished the sixteen rounds. The result is latched into IF. Simultaneously, a new 64-bit block of data is loaded into LR.
- 33–40: While DES starts the second cipherment, IF outputs c_0 . The scheduling scheme goes on, with a periodicity of 16 clock cycles.

In practice, the pipeline is connected to a scratch-pad RAM. The pipeline reads from (cycles 1–8, 9–16, 25–32) and writes to (cycles 17–24, 33–40) the RAM on disjoint time slots. Therefore, a simple-port RAM (the less expensive type of RAM) is perfectly suitable. The throughput of the DES pipelined operator is 64-bit per 16 clock periods (4.0 bit/clock). The input and output latencies are 8 cycles (as in Sec. 3.2.2.1, we ignore the key initial loading.)

By the same token, the pipelined architecture improves the datapath speed. In the straightforward implementation, the LR register has four input sources:

1. the input byte concatenated with the previous register content shifted by 8 bits to build the plaintext up,
2. the same block, but passed through IP, to start the computation and
3. the end of the round data, reinjected into LR for the next round.
4. the same block, swapped and passed through FP.

As already shown in Fig. 3.2, a $4 \rightarrow 1$ multiplexor, to choose between those four sources, directly precedes LR.

In the pipelined architecture, IP is performed concomitantly with the collection of the plaintext constitutive bytes. It does not slow down the computation, because in a hardware implementation, IP requires no logic: it is a mere reordering of wires. Consequently, LR has only two possible inputs in the pipelined architecture ; the $4 \rightarrow 1$ multiplexor is replaced by a $2 \rightarrow 1$. This optimization is crucial, since this multiplexor is on the critical path (LR $\rightarrow$ Round Logic $\rightarrow$ LR).

3.2.3 Optimal software / hardware partition to realize all DES variants

3.2.3.1 IP representation

The notations used in this section are inspired from `openssl` [30] internals:

- `des_encrypt1` is the full DES,
- `des_encrypt2` $\doteq$ IP $\circ$ `des_encrypt1` $\circ$ FP
is DES, without IP nor FP.

Functions `des_encrypt{1,2}(m, k, enc)` take three arguments: a message m , a key k and a Boolean enc , specifying whether to encrypt ($enc = 1$) or decrypt ($enc = 0$.)

For any function set $f_i : [0:2^{64} - 1] \mapsto [0:2^{64} - 1]$, the following property holds:

$$\Pi_i (\text{FP} \circ f_i \circ \text{IP}) = \text{FP} \circ (\Pi_i f_i) \circ \text{IP}, \quad (3.1)$$

$$\text{where: } \Pi_{i=i_{\min}}^{i=i_{\max}} f_i \doteq f_{i_{\max}} \circ \dots \circ f_{i_{\min}},$$

because $\text{FP} \circ \text{IP}$ is the identity function.

This property allows the chaining of DES operations without caring neither for IP nor for FP permutations. The “IP representation” computational framework consists in using the `des_encrypt2` primitive instead of `des_encrypt1`, the IP (resp. FP) being called only once at the beginning (resp. at the end) of the computation. The Equation (3.1) can be applied to the following DES variants:

- $f_i = \text{des_encrypt2}(m_i, k, enc)$ (ECB and ECB^{-1})
- $f_i = \begin{cases} \text{des_encrypt2}(m_i, k_i, (1+i) \% 2) & \text{if } enc = 1 \\ \text{des_encrypt2}(m_i, k_{2-i}, (i) \% 2) & \text{if } enc = 0 \end{cases}, \forall i \in \{0, 1, 2\},$
(triple-DES on one block m ; $m_0 = m$ and $m_{i+1} = f_i(m_i)$, the output being m_3)
- $f_i = \begin{cases} \text{des_encrypt2}(m_i \oplus f_{i-1}, k, 1) & \text{if } enc = 1, \\ \text{des_encrypt2}(m_i, k, 0) \oplus f_{i-1} & \text{if } enc = 0, \end{cases}$
(CBC and CBC^{-1} , with $f_{-1} = \text{IV}$)
- $f_i = \text{des_encrypt2}(f_{i-1}, k, 1) \oplus m_i,$
(64-CFB and 64-CFB^{-1} , with $f_{-1} = \text{IV}$)
- $f_i = \text{des_encrypt2}(f_{i-1} \oplus m_{i-1}, k, 1) \oplus m_i,$
(64-OFB and 64-OFB^{-1} , with $f_{-1} \oplus m_{-1} = \text{IV}$)

In software implementations, IP is not free as in hardware, because bits cannot be arbitrarily moved within or between words. In `openssl`, IP and FP are implemented using 32-bits registers in $5 \times (3 \text{ XOR} + 2 \text{ SHIFT} + 1 \text{ AND}) = 30$ operations [30].

DES `des_{en,de}crypt3` function performs triple DES on one block of plaintext. It is the only function from `openssl` that takes advantage of the optimization provided by the computation in the IP representation (3.1). All other functions, especially chained DES, are thus inefficient.

3.2.3.2 Multi-mode pipelined DES datapath operating in “IP representation”

The pipeline described in Sec. 3.2.2.2 (see Fig. 3.4) is not designed to chain ciphersments. However, it can be enhanced to cope with triple-DES and all modes of operation. The rationale is to add two inputs to the LR multiplexer:

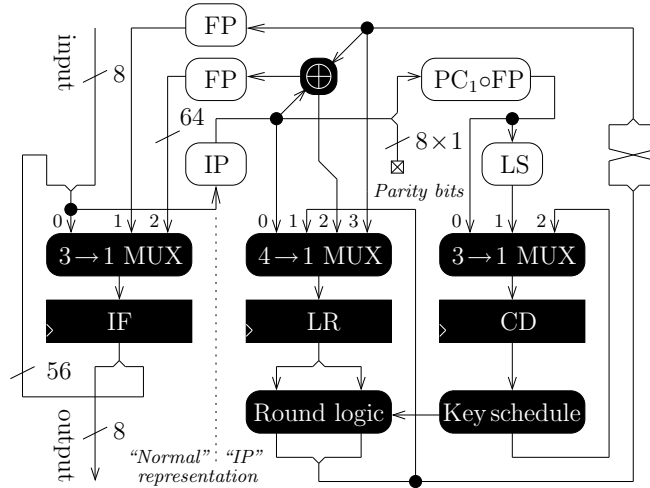


Figure 3.6: Proposed multi-modes pipelined DES datapath operating in “IP representation”.

Table 3.2: Selected signals at the beginning of each DES chained with modes of operation.

Mode	IF MUX	LR MUX	Built upon
ECB	1	0	DES
ECB ⁻¹	1	0	DES ⁻¹
CBC	1	2	DES
CBC ⁻¹	—	—	—
CFB	2	2	DES
CFB ⁻¹	2	0	DES
OFB = OFB ⁻¹	2	0, 3, 3, ...	DES

1. the result of the previous DES, which allows triple-DES and also OFB (where the series $\{\text{DES}^i(\text{IV})\}_{i \geq 0}$ is computed),
2. *idem*, but XORed with the new message, which allows CBC and CFB chained modes.

The new inputs to LR are compatible provided they are in the IP representation. It basically means that inputs to DES must be previously IP'ed and that output of DES to be recycled must not be FP'ed. Additionally, the IF register must be able to latch the XOR between the new message and the current result, which is required by the stream modes (*i.e.* CFB and OFB) of DES. Those constraints lead to the versatile version of the pipelined DES datapath represented in Fig. 3.6. By default, the multiplexor in front of IF (resp. LR) selects the input 0 (resp. 1). At the end of every cipherment (*i.e.* every 16 clock periods), the multiplexers choose another input, as shown in Tab. 3.2.

The realization of triple DES requires a special schedule. The 3DES-ECB is illustrated in Fig. 3.7. The IF and CD registers sample their default inputs, selection 1 for IF and 0 for CD (corresponding to the ECB and ECB⁻¹ lines in Tab. 3.2). The scheme for 3DES of Fig. 3.7 can be combined with the modes of operation. It suffices that the data to be output by IF and sampled into LR have non-default origins documented in Tab. 3.2 every 3×16 clock periods.

In the case of 3DES with two keys (k_0 and k_1 , $k_2 = k_0$), it is noticeable that the computation never stalls. As a matter of fact, the key for the first of the three DES is already present in

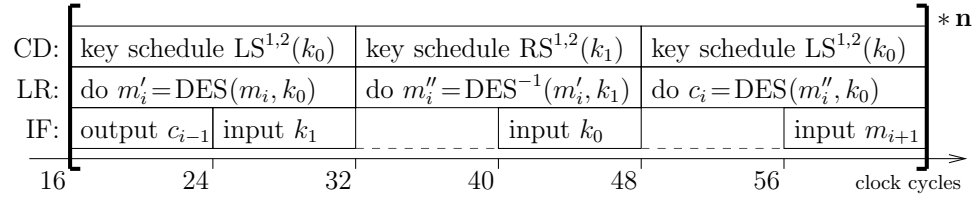


Figure 3.7: Register contents when the pipeline is configured for 3DES encipherments with two keys k_0 and k_1 , possibly chained $i \in [0 : n[$ times (in which case the indicated clock cycles must be added the offset $i \times 48$.)

CD, since the last key was $k_2 = k_0$. Consequently a new message m_i can be loaded instead, and the next computation can follow seamlessly.

3.2.3.3 SW/HW trade-offs

The proposed pipelined architecture of Fig. 3.6 is versatile, since all the modes of operation can be fit. Nevertheless, this architecture suffers three drawbacks, discussed in the following three paragraphs.

3.2.3.3.1 Realization of 3DES with three different keys.

In 3DES with three keys, it would be necessary to load the first key k_0 and the new message block m_i at the same time. However, in the proposed versatile architecture, the RAM delivering the data is single-port and there is a single IF register. As there is a contention, the two loadings must be done sequentially. As CD has kept a key globally unchanged during 8 clock cycles, it is loaded first. During the extra eight clock cycles required to load m_i , the pipeline stalls, because it is starving data. Triple-DES with three keys can thus be used with modes of operation, but it is the only exception where the cipherments do not chain gracefully.

3.2.3.3.2 Realization of CBC^{-1} .

As already indicated in Tab. 3.2, CBC cannot be deciphered directly. The reason is that to retrieve plaintext block m_i , the following XOR must be computed: $m_i = \text{DES}^{-1}(c_i) \oplus c_{i-1}$. Unfortunately, the XOR right-hand side c_{i-1} has already been consumed by the pipeline (to compute $\text{DES}^{-1}(c_{i-1})$) when it is needed again. Re-fetching the ciphertext c_{i-1} in memory would require to freeze the pipeline during 8 clocks cycles, which is not desirable.

The first workaround is to implement CBC^{-1} by EBC^{-1} , which yields $m_0, m_0 \oplus m_1, m_1 \oplus m_2, \text{etc.}$ instead of $m_0, m_1, m_2, \text{etc.}$ The processor can afterwards compute (in software) the XOR between the couples in the memory $\text{ram}[0:N[$ to retrieve the correct plaintext. An example programme is listed below:

```

register char tmp0, tmp1;
for( register char i=0; i<8; ++i ) {
    tmp0=ram[i]; // The 1st block is only read
    for( register size_t j=1; j<N; ++j ) {
        tmp1=ram[j*8+i]; // Read jth block
        ram[j*8+i]=tmp0^tmp1; // Write jth block
        tmp0=tmp1;
    }
}

```

The second workaround we propose is the smartest, because it does not require any post-processing in software. The idea is to adapt the control to decipher the blocks c_i in reverse

Table 3.3: Resources area and maximum frequency of the three proposed architectures implemented in a Xilinx VIRTEX-4.

Architecture	Number of CLBs	Number of DFFs	Frequency
“Straightforward” (Fig. 3.2)	1445	209	211 MHz
“Pipelined” (Fig. 3.4)	1454	259	202 MHz
“Multi-mode” (Fig. 3.6)	1957	276	144 MHz

order. If we note $c'_i \doteq c_{N-1-i}$, then $m_i = \text{DES}^{-1}(c'_{i-1}) \oplus c'_i$, for $i \in]N:0]$, is computable by the multi-mode architecture of Fig. 3.6. It is the same configuration as CFB^{-1} , but with the key schedule set to decipher.

3.2.3.3.3 Using CBC and CBC^{-1} with an IV.

At last, CBC and CBC^{-1} modes cannot be used with an IV. The IV should indeed be loaded, kept in some register (say LR) while the first block m_0 is built-up into IF. The computation could then start with the first operand $\text{IV} \oplus m_0$. However, this scenario also implies that LR has an enable, which we explicitly want to avoid.

A first solution relies on the software. The task simply consists in XORing the first block prior to calling an encipherment or after a decipherment.

A second solution consists in adding an initialization procedure, during which $\text{DES}^{\pm 1}(\text{IV})$ is computed. Then, every message to cipher is simply prepended $\text{DES}^{\pm 1}(\text{IV})$. For long messages, this overhead in processing time becomes negligible.

A third solution implies to increase the DES engine area. The datapath is augmented with an 8-bit XOR operator that would compute “input $\oplus$ output” (with the notations of Fig. 3.6.) This result would be injected into the multiplexor in front of the IF register. It is a design choice to decide whether it is worth implementing this minor hardware feature that complexifies both the datapath and the control (since the IF multiplexor has a new input).

3.2.4 Performance evaluation of the proposed architecture

The three architectures discussed in this section, namely the “straightforward” (Fig. 3.2), “pipelined” (Fig. 3.4) and “multi-mode” (Fig. 3.6) have been captured using VHDL. They have been synthesized in an FPGA technology (for prototyping) and in an ASIC “low-leakage” 130 nm technology (for production.)

The FPGA front-end was Mentor Graphics Precision Synthesis (for *mapping*) and the back-end Xilinx ISE (for *fitting*.) The performances are given in Table 3.3 for the VIRTEX 4VFX12SF363-12.

In terms of speed, the “straightforward” architecture is the fastest and the “multi-mode” is the slowest.

The ASIC tool-chain for the tape-out of the embedded 8-bit DES blocks was Cadence `pks_shell` for the front-end (synthesis) and `SOC/Encounter` for the back-end (place-and-route.) The synthesis results, for both the control and the datapath, are given in Fig. 3.8. The increase of the area is due for about 25 % to bufferization and for 75 % to Boolean optimization. The straightforward architecture is the most compact and the multi-mode is the largest. The designs maximum frequency are 540 MHz (straightforward DES), 500 MHz (pipelined DES), 435 MHz (multi-mode DES.) The pipelined DES does not reach the same frequency as the straightforward DES because its more complex control limits its speed. The multi-mode DES datapath is more sophisticated, which explains why it cannot reach as high frequencies as the two other architectures. The maximum frequency of the proposed architectures are fairly high

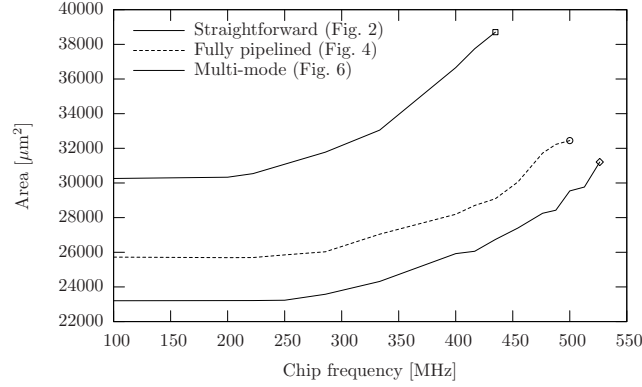


Figure 3.8: Synthesis results for the three architectures.

Table 3.4: Throughput in bit/cycle of some modes of the three studied implementations of DES.

	Straightforward	Pipelined	Multi-mode
DES-ECB	2.000	4.000	4.000
DES-CBC	1.000	1.000	4.000
3DES-ECB	0.571	0.571	1.333
3DES-CBC	0.444	0.444	1.333

for an embedded system. The architectures can be adapted to an external datapath width of $n = 16$ (resp. $n = 32$) bits, in which case two (resp. four) rounds can be computed within one clock period. This new architecture will run at a maximum speed roughly half (resp. four times less.)

However, the cipherment throughput is the highest for the pipelined architecture in ECB mode, and for the multi-mode in all the other modes and triple DES. Table 3.4 shows the throughput of some modes. It should be noted that neither the straightforward nor the pipelined architectures are designed to handle modes of operation or triple-DES. The chaining operation must thus be computed artificially in SW. An estimation of the code for such an operation is given below:

1. Read `ram[i]` (1 clock cycle)
2. Read `ram[i+8]` (1 clock cycle)
3. Compute `ram[i] XOR ram[i+8]` (1 clock cycle)
4. Write `ram[i+8]` (1 clock cycle)

This fragment must be repeated 8 times, which leads to a total of $8 \times 4 = 32$ clock cycles. This evaluation is optimistic, because it does not take into account the context switch. It is also unrealistic, since the processor should not be disturbed by the computation internal details. The throughput figures given for straightforward or the pipelined are thus only indicative.

The maximum throughputs are also shown graphically in Fig. 3.9 (a) for DES-ECB and in Fig. 3.9 (b) for 3DES-CBC. It is also interesting to compare the throughputs of an ASIC design with the one of a personal computer (PC.) The maximum throughput for 3DES-CBC attained

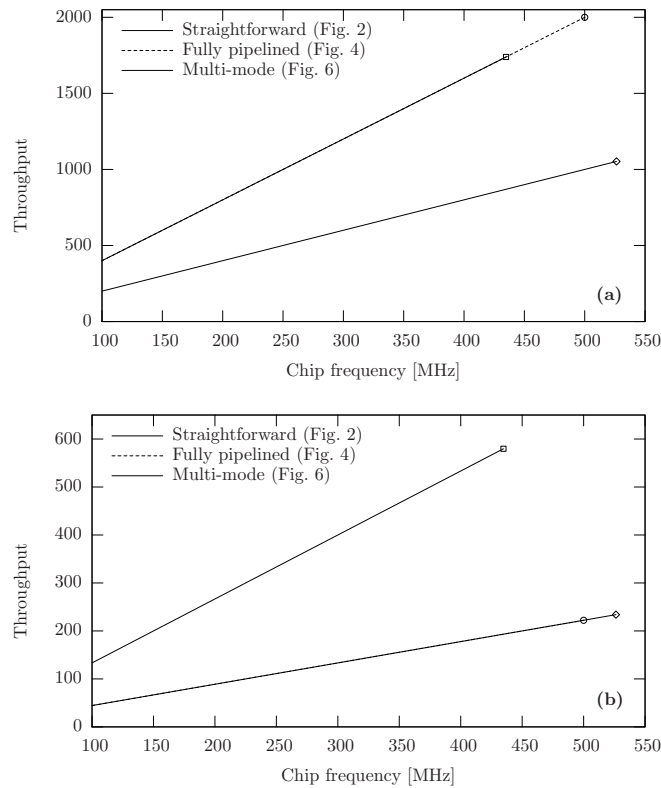


Figure 3.9: Throughput (in 10^6 bit/s) of the three solutions in (a) DES-ECB and (b) 3DES-CBC.

by the multi-mode architecture is $580 \text{ Mbit/s} = 435 \text{ MHz} \times 1.33 \text{ bit/cycle}$ while a PC clocked at 3.2 GHz is only able to encrypt at 200 Mbit/s (result of `openssl speed des` [30].)

However, achieving high throughput would be needless if the area overhead is getting too large. For most modes of operation, the parallelization of the ciphersments is impossible, due to data dependencies between the consecutive blocks. Still, $\text{ECB}^{\pm 1}$, CBC^{-1} and CFB^{-1} can indeed be parallelized. In those cases, the throughput can be multiplied by the instantiation of multiple engines operating concurrently. Therefore, in Fig. 3.10, the throughput divided by the area is plotted. At constant area, the multi-mode architecture of DES remains the fastest.

The DES module after automatic place-and-route by **SOC/Encounter** is shown in fig. 3.11. It happens that the synthesizer was optimistic: static timing analysis performed on the final layout at 95% placement density reports, after post-route resynthesis and in-place optimization, a maximal frequency of 286 MHz (*versus* 435 MHz predicted by the logical synthesizer.) This limitation is in practice not deterrent, since 256 bytes embedded rams in 130 nm technology cannot work above 333 MHz without violating either hold or setup times.

3.2.5 Comparison with other fast and versatile implementations of DES

A “Cryptographic Reuse Library” based on static genericity is described in [91]. It contains synthesizable algorithms commonly used in cryptography, each of which can be used either as such, or wrapped into a module that enables modes of operations, or further wrapped into

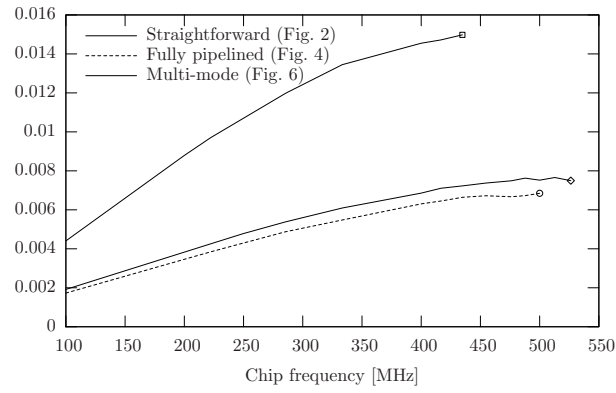


Figure 3.10: Comparative efficiency of 3DES-CBC (in $\text{Mbit/s}/\mu\text{m}^2$) for the three proposed architectures.

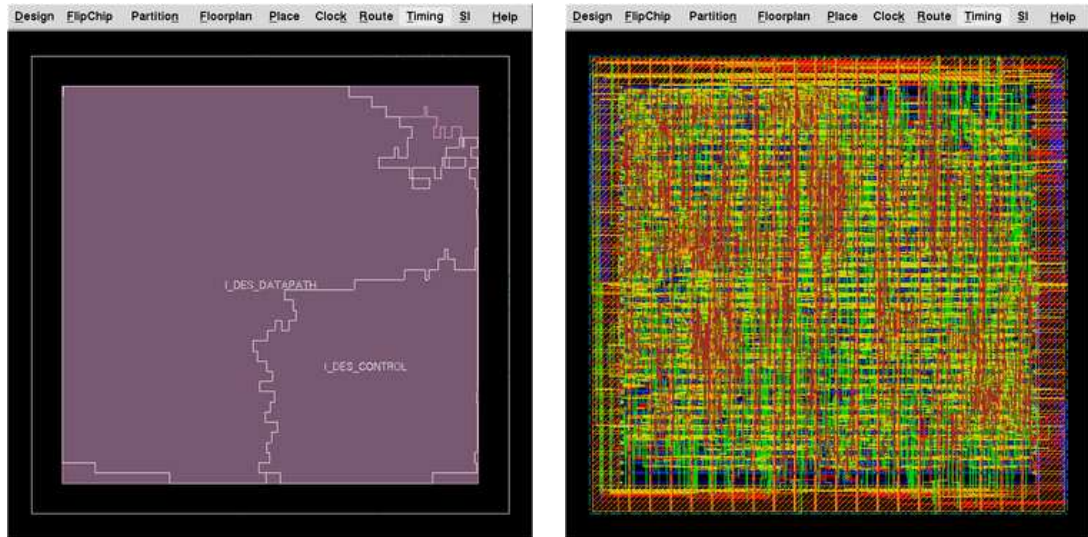


Figure 3.11: The multi-mode DES after place-and-run in 130 nm technology. *Left*: Datapath / Control partitioning; *Right*: Final layout.

an interface module that adapts throughputs and latencies to match that of the environment. Although the methodology has not been applied to DES in [91], it could be extended to support this algorithm. The features of this “Cryptographic Reuse Library” are those we present in this section. However, as the mode of operation and interface wrappers involved in the library are not aware of the algorithm internals, the resulting block is necessarily sub-optimal. The approach used in the “multi-mode” architecture (Fig. 3.6) is to merge the two abovementioned wrappers into the algorithm datapath itself. This allows the “multi-mode” architecture to work without dead cycles at a constant throughput. This prominent feature is a valuable characteristic of the multi-mode architecture: the I/Os are equipartitioned during the processing of the DES algorithm. However, this design solution is specific to DES, and probably does not extend to other algorithms.

Some architectural innovations are described in [36] regarding the round logic of DES. The frontier between the consecutive rounds i and $i + 1$ is dissolved in order to balance the critical path between $L_i \rightarrow R_{i+1}$ and $R_i \rightarrow L_{i+1}$. The transformation yields an overall decrease of the critical path length, at the cost of an increase of the latency (the apparent number of rounds rises from 16 up to 21 or 37) and of a particularization of the first and last rounds. These modifications are not a burden when a pipelined implementation is targeted. However, they are deterrent for the architectures presented in this section, because the data processing is kept iterative.

3.2.6 Proposed architectures modifications for bit-slice P&R

The three architectures proposed in figures 3.2, 3.4 and 3.6 are not adapted to a bit-slice place-and-route. The permutations IP, FP and PC1 indeed mix the bus indices, which wastes area simply for wires reordering. A reorganization of the permutations that helps keeping buses together is shown in Fig. 3.12. The figure highlights the various datapath bit-widths and introduces three permutations $X\{1,2,3\}$. $X1$ and $X2$ correspond to IP and FP. Those permutations take advantage of the fact that IP and FP can be achieved in eight cycles (duration of a 64-bit block loading or unloading.) As IP and FP are close to being transpositions of the input bits represented as an 8×8 matrix (see Fig. 3.13), the operation can be realized in eight iterations thanks to shift registers (see Fig. 3.14.) The permutation $X3$ remains the adaptor from IP to PC1: $X3 \doteq PC1 \circ FP$.

Notice that the multiplexor MUXPC2 is placed after register CD in the architecture of Fig. 3.12. This is a design awkwardness, because the key schedule is uselessly added to critical path. The ASIC “SecMat V1” available for the power analysis has this very architecture, which is why the figure is not corrected. All the power traces on DES printed in this document were realized on this architecture.

3.2.7 Conclusion on the DES architecture

Two architectural innovations, namely the I/O and processing pipelining and the use of the “IP representation”, allow to improve the design of DES 8-bit implementations. The proposed architecture supports all modes of operation and triple DES with two keys. The hardware implementation can take advantage of both methods, whereas software implementations can only benefit from the “IP representation”. The pipelining strategy consists in parallelizing the data inputs and outputs with the processing. It also enables shorter clock periods, due to the elimination of the some multiplexors on the critical path. The IP representation enables optimized chaining. These optimizations allow to accelerate DES operations in smartcards or in embedded systems or to speed-up DES-cracking machines [81].

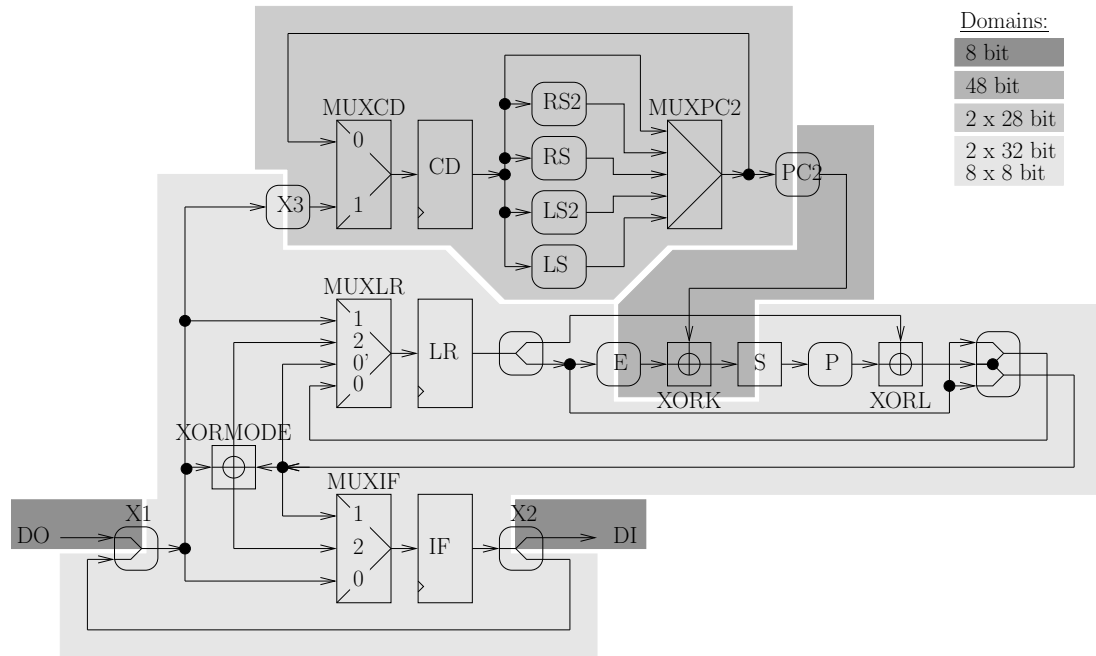


Figure 3.12: SecMat V1 version of the DES datapath, with the indication of the datapath width (in shades of gray.)

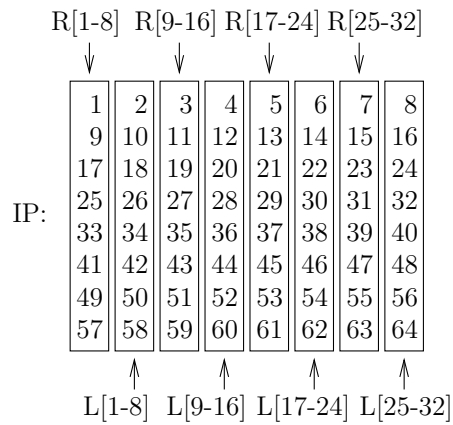


Figure 3.13: Repartition of the messages bytes (lines) in registers L and R after IP (columns.)

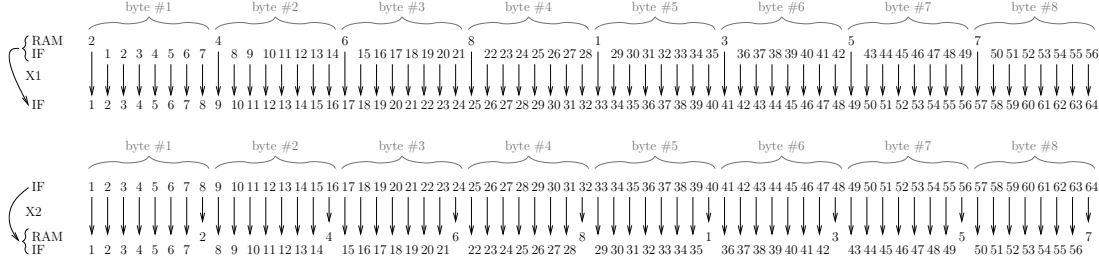
Figure 3.14: Permutations X1 (*top*) and X2 (*bottom*) used in the DES datapath of Fig. 3.12.

Table 3.5: Number of CD shifts for the combinatorial DES datapath (see Fig. 3.15.)

Round #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shift #	0	1	3	5	7	9	11	13	14	16	18	20	22	24	26	27

3.3 Fully combinatorial DES implementation

3.3.1 Combinatorial DES datapath

The iterative architecture presented in the previous section can be made combinatorial, without modifying neither the control nor the performances in terms of speed. The idea is to remove the register transfers occurring during the sixteen rounds. The registers are thus disabled during the rounds execution, and the datapath is fully unrolled. The algorithm combinatorial depth is thus roughly increased by a factor of sixteen, but the registers LR and CR remain frozen during sixteen clock cycles, which makes up for the delay through the gates. The architecture is depicted in Fig. 3.15. The inputs 1 of the LR multiplexor and 2 of the CD multiplexor play the role of enable for the corresponding registers. The key schedule consists in the sequence of shifts (left as for encryption, right otherwise) given in Tab. 3.5.

Notice that, at the synthesis, the timing constraints must be relaxed. Indeed, by default, the timing engines of the synthesizers attempt to fit into one clock cycle the logic that is situated between two register banks. In the combinatorial DES specific case, the logic driven by LR and CD has sixteen clock cycles to execute. This piece of information cannot be easily inferred, thus user constraints must be set. They basically consist in specifying spare clock cycles for some timings arcs. The timing paths that are concerned thus start at registers LR and CD, plus the Boolean signal originating from the control that tells whether the current operation is an encipherment or a decipherment (refer to Fig. 3.5, where the shifts can be interpreted left or right-wise.) The “multi-cycle” constraints listed in Fig. 3.16 express the fact that outputs of LR and CD are sixteen times slower than the clock and that the signal to decide between encipherment and decipherment is a false timing path. This last path is indeed never critical because the choice between encryption and decryption is not modified during one computation.

Finally, the description in HDL is impacting the synthesis of the combinatorial datapath. With the state-of-the-art synthesizers, it is not possible (within a reasonable amount of time) for a synthesizer to map the function: $LR_{16} = IP \circ DES(IP^{-1}(LR_0), PC1^{-1}(CD_0))$. It happens that the synthesizer is able to cope with an unrolled description expressed in the generic instantiation of sixteen entities, as shown in Fig. 3.17.

This “segmented” description yields thus the best netlist obtainable with general-purpose synthesizers. A dedicated Boolean mapping tool might be more efficient. With the round-by-round description, the synthesizer can optimize the round entities boundaries, but not their

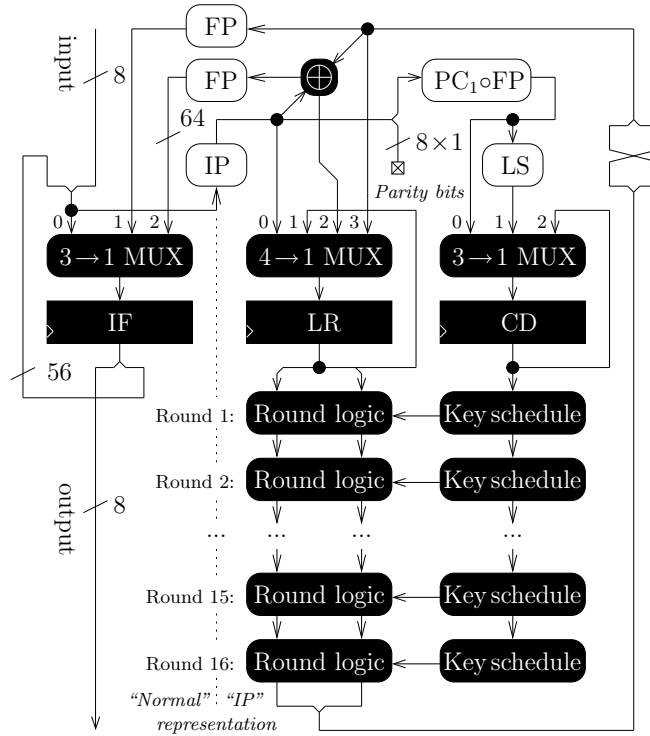


Figure 3.15: Combinatorial DES datapath built upon the versatile architecture of Fig. 3.6.

```

set_current_module des_datapath_combi_wrapper; # Internal constraints
set_current_instance [find -hier -inst I_REG_LR];
# The following constraint (1+15 cycles allowed for the computation)
# concerns the whole bus:
set_cycle_addition -from [get_info [lindex [find -port q] 0] bus] 15;
set_current_instance [find -hier -inst I_REG_CD];
set_cycle_addition -from [get_info [lindex [find -port q] 0] bus] 15;
set_current_module des_datapath_combi; # External constraint
set_false_path -from [find -port sel_left_not_right]; # Encrypt/Decrypt

```

Figure 3.16: TCL timing constraints crafted for the “multi-cycle” DES combinatorial datapath synthesis by Cadence `bgx_shell`.

```

G_ROUND: for ROUND in 1 to 16 generate
  I_ROUND: entity des_datapath_combi_round( combi )
    port map
    (
      M      => LR( ROUND-1 ),
      C      => LR( ROUND  ),
      KEY    => reg_cd_out,
      CYPHER => sel_left_not_right, -- Encrypt/Decrypt
      ROUND  => ROUND
    );
end generate G_ROUND;

```

Figure 3.17: Excerpt from the VHDL description of the unrolled DES architecture.

internal structure. As a consequence, the iterative nature of the DES datapath is still somehow structurally present.

3.3.2 Security properties of the combinatorial DES

The expected security enhancement provided by the combinatorial DES first arises from the removal of the registers: there are consequently no more register transfers to exploit. As underlined in Sec. 2.4.2, the attacks of combinatorial logic is more difficult to set up and, above all, features a reduction of the SNR by a factor two (refer to Eqn. (2.16).) The absence of DFFs to resynchronize the combinatorial signals at every round fosters the timing variations to increase during the percolation of the message and of the key. A statistical study of the ciphertext arrival time variations is given below.

While generating the netlist of the combinatorial datapath, the propagation delays in the gates as well as an estimation of the routing delays are saved in a back-annotation SDF [5] file. A simulation is then run with the VITAL [48] models of the gates. It must be noted that the simulation was performed on a DES module with IP and FP.

For 1251 different random cleartexts, three values are extracted:

1. the date of the first toggle at the output LR_{16} of the datapath,
2. the date of the last toggle, corresponding to the bit final value, and
3. the number of toggles, corresponding to the “glitch” activity of the output bit.

The average values of these three measurements are plotted on the three figures 3.18 (a), (b) and (c). The standard deviation of the extracted quantities is computed and reported on the graphs as error bars.

The following observations can be made:

1. **First toggle:** The first event in output occurs quickly compared to the computation duration. In average, the first toggle occurs 2 ns after the computation beginning for an average duration of 15 ns. This means that there exists very short paths to the output, as well as very long ones. Thus we expect many glitches.
2. **Last toggle:** The output bits can be partitioned in two groups L_{16} and R_{16} . Given the structure of FP, at the output of the full combinatorial DES, the odd index bits all come from R_{16} while the even index bits all come from L_{16} . Now, because of the Feistel-network structure, the evaluation of L_{16} is immediate, whereas R_{16} must first pass through the round logic. The computation time difference is about equal $1/16$ of the total computation

time. Additionally, the standard deviation on the last toggle date is equal to one round duration, which proves that it is extremely difficult for an attacker to resynchronize one output bit. The instant when a relevant event occurs is indeed mixed with the events from the previous round, as suggested by the interpenetration of the error bars from the two groups of output bits.

3. **Number of toggles:** The number of glitches on the outputs is, in average, equal to 20, but is very scattered. The dispersion is a characteristic of the netlist, because it is little impacted by a change of the key. This tends to show that the fully combinatorial DES remains immune to *timing attacks* [53]. But this immunity is rather a property of the algorithm ; a timing attack can be mounted successfully against an unprotected implementation of RSA because every bit of the secret key is processed sequentially and because the duration of any single multiplication depends on this bit. As far as DES is concerned, the secret bits are used in parallel and their addition (bitwise XOR) operates in constant time (at the first order.) The parallel processing and the absence of obvious distinguishers in execution time make the fully combinatorial DES implementation robust against timing attacks.

3.4 DES remarkable cryptological properties

The DES cipher has several cryptological properties that are exposed in this section 3.4 (*theory*.) The application of those properties to side-channel analysis is the subject of the next section 3.5 (*practice*.)

DES is a Feistel scheme with sixteen rounds. One typical Feistel round is depicted in Fig. 3.19.

As for DES, the Feistel function f is constant, and thoroughly studied in the previous section 3.2. The round key K_i depends on the round i . A DES key is first stripped off its parity bits thanks to a $64 \rightarrow 56$ bit function called PC_1 . The result is stored as a key state, named CD, each register C and D holding 28 bits. The state CD undergoes transformation referred to as “Left-Shifts” (LS) or “Right-Shifts” (RS), depending on the type of cipherment (*encryption* or *decryption*.) A remarkable property of the DES key schedule is that after one cipherment, the register CD is back to its original state. The round key K_i injected into the datapath is the state CD filtered through a function called PC_2 . The PC_2 function does not make any use of bits {9, 18, 22, 25, 35, 38, 43, 54} of CD. The state of CD (*i.e* the key before PC_2) is given in Tab. 3.6.

The Data Encryption Standard has a couple of remarkable properties, that constitute potential weaknesses:

1. **Complementary property:**

$$\forall (K, M) \in \{0, 1\}^{56} \times \{0, 1\}^{64}, \quad \text{DES}_{\overline{K}}(\overline{M}) = \overline{\text{DES}_K(M)}.$$

2. **Weak-keys and fixed points** (refer to next section 3.4.1.)
3. **Semi-weak keys and anti-fixed points** (refer to next section 3.4.2.)

No cryptographic engineering handbook (neither [60, Sec. 7.4.3] nor [22]) provide demonstrations for DES particular keys, therefore I give here mine. A more elliptic proof can be found in [63]. They will be useful for the coming power analyzes of DES (see Sec. 3.5.)

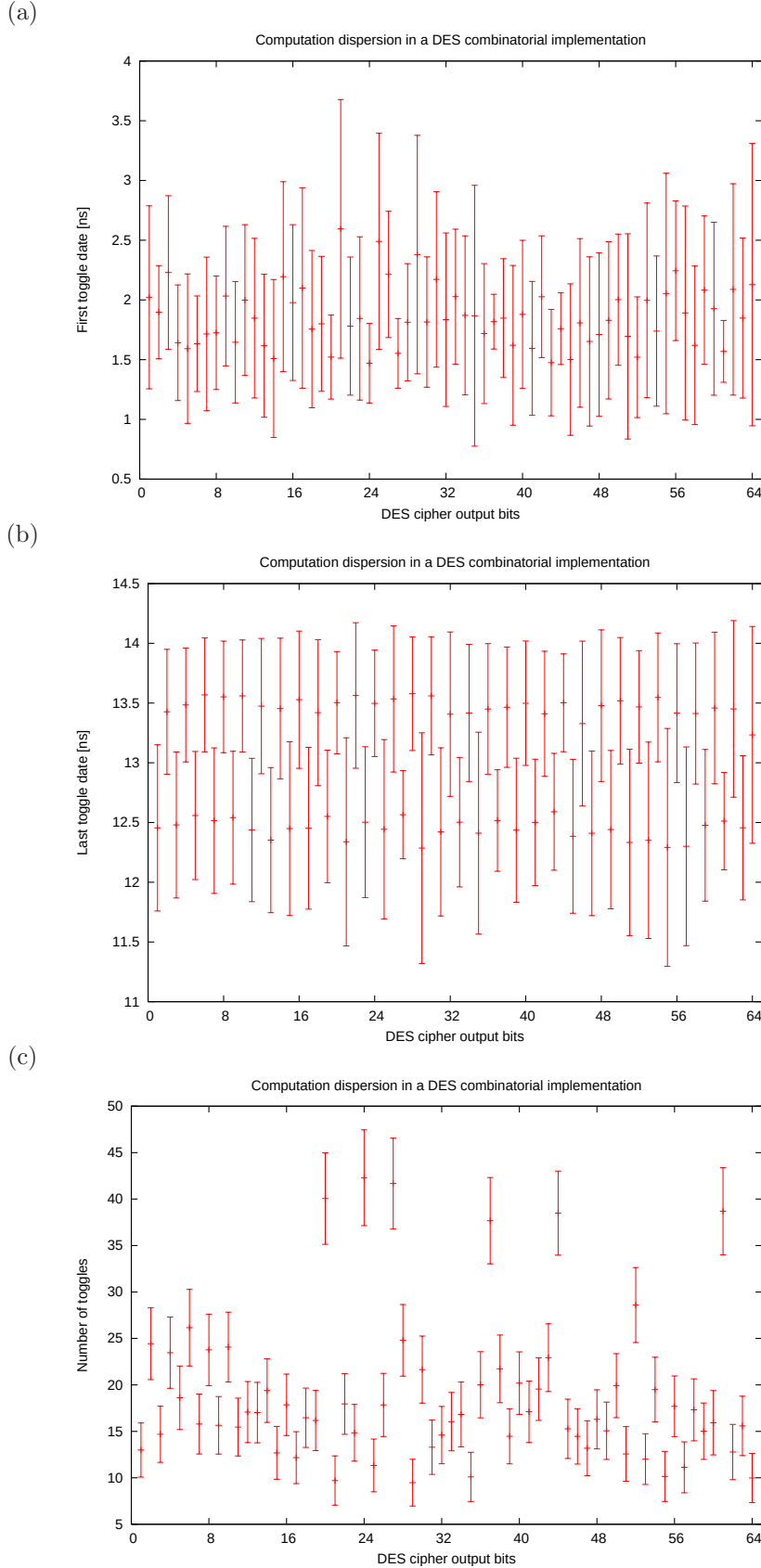


Figure 3.18: Statistics on a back-annotated combinational DES netlist simulation with the key 0x01 23 45 67 89 ab cd ef: (a) first toggle date, (b) last toggle date and (c) number of toggles.

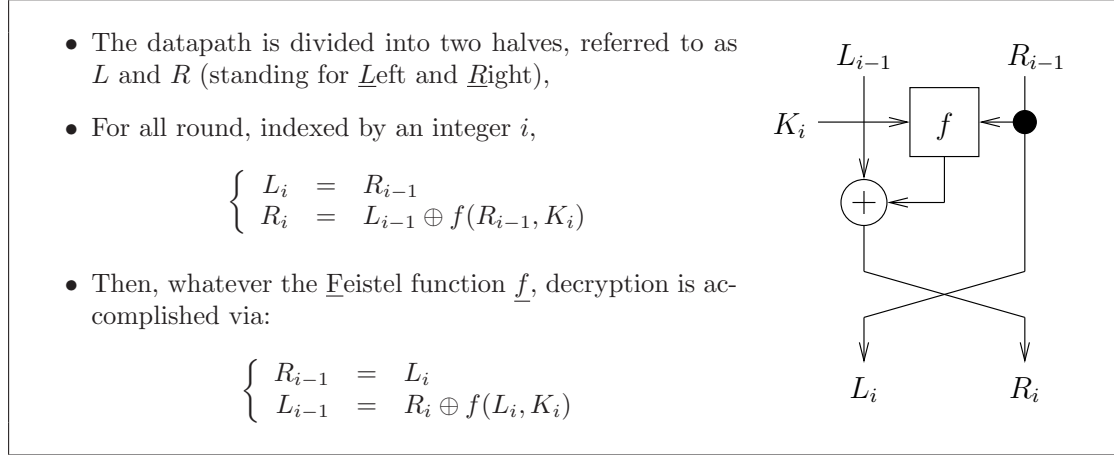


Figure 3.19: Horst Feistel network structures [45].

Table 3.6: DES architecture V1 key schedule.

Round #	Encryption		Decryption	
	# LSs	CD State	# RSs	CD State
1	1	LS ¹	0	RS ⁰ = Id
2	1	LS ²	1	RS ¹
3	2	LS ⁴	2	RS ³
4	2	LS ⁶	2	RS ⁵
5	2	LS ⁸	2	RS ⁷
6	2	LS ¹⁰	2	RS ⁹
7	2	LS ¹²	2	RS ¹¹
8	2	LS ¹⁴	2	RS ¹³
9	1	LS ¹⁵	1	RS ¹⁴
10	2	LS ¹⁷	2	RS ¹⁶
11	2	LS ¹⁹	2	RS ¹⁸
12	2	LS ²¹	2	RS ²⁰
13	2	LS ²³	2	RS ²²
14	2	LS ²⁵	2	RS ²⁴
15	2	LS ²⁷	2	RS ²⁶
16	1	LS ²⁸ = Id	1	RS ²⁷

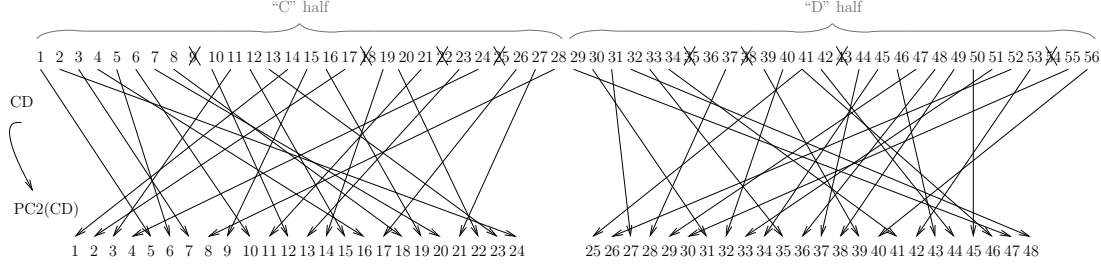


Figure 3.20: The PC2 *permuted choice*, operating from $\{0, 1\}^{28} \times \{0, 1\}^{28}$ to $\{0, 1\}^{24} \times \{0, 1\}^{24}$.

3.4.1 DES weak keys and fixed points

3.4.1.1 DES weak keys

Definition 2 A *weak key* K is a key such as: $\forall M, DES_K(M) = DES_K^{-1}(M)$. As a consequence, encryption and decryption are *involution*: $DES_K \circ DES_K = DES_K^{-1} \circ DES_K^{-1} = Id$.

Property 1 Note: For the sake of simplicity, in the following, $K = PC1(key)$.

- The key schedule must be **palindromic**. $\forall i \in [1, 16], K_i = K_{17-i}, PC2(CD_i) = PC2(CD_{17-i})$.
- $PC2: \{0, 1\}^{56} \rightarrow \{0, 1\}^{48}$ is **not bijective**;
 CD_i and CD_{17-i} are equal only on $[1, 56] \setminus (\{9, 18, 22, 25\} \cup \{35, 38, 43, 54\})$.
- As shown below, **all the subkeys are equal**.
- Moreover, the subkeys (all equal to the actual key K) take only **four values**:
 $\{0\}^{56}$, $\{0\}^{28} \times \{1\}^{28}$, $\{1\}^{28} \times \{0\}^{28}$ and $\{1\}^{56}$.

Demonstration 2 (Proof of property 1) As shown in Fig. 3.20, PC2 can split into two halves $[1, 28] \cup [29, 56]$. Thus solutions can be thought of independently for “C” and “D”. In the sequel, only the “C” part is discussed.

- For $i = 1$, $CD_1 = LS(K)$ and $CD_{16} = K$. Thus:

$$\begin{array}{l}
 LS(K) : 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ \cancel{10} \ 11 \ 12 \ 13 \ 14 \ 15 \ 16 \ 17 \ 18 \ \cancel{19} \ 20 \ 21 \ 22 \ \cancel{23} \ 24 \ 25 \ \cancel{26} \ 27 \ 28 \ 1 \\
 \quad \quad \quad || \quad || \quad || \quad || \quad || \quad || \quad || \quad \text{?} \quad || \quad || \quad || \quad || \quad || \quad || \quad || \quad \text{?} \quad || \quad || \quad || \quad || \quad \text{?} \quad || \quad || \quad || \\
 Id(K) : 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ \cancel{9} \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16 \ 17 \ \cancel{18} \ 19 \ 20 \ 21 \ \cancel{22} \ 23 \ 24 \ \cancel{25} \ 26 \ 27 \ 28
 \end{array}$$

Hence the bits of the key K are equal in every of the four partitions: $([26, 28] \cup [1, 9])$, $([10, 18])$, $([19, 22])$ and $([23, 25])$. Note: The symbol “||” is used to express the equality between bits of two bit vectors, whereas “?” means that no equality relationship holds.

- For $i = 2$, $CD_2 = LS^2(K)$ and $CD_{15} = LS^{27}(K) = RS(K)$. Thus:

$$\begin{array}{l}
 LS^2(K) : 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ \cancel{11} \ 12 \ 13 \ 14 \ 15 \ 16 \ 17 \ 18 \ 19 \ \cancel{20} \ 21 \ 22 \ 23 \ \cancel{24} \ 25 \ 26 \ \cancel{27} \ 28 \ 1 \ 2 \\
 \quad \quad \quad || \quad || \quad || \quad || \quad || \quad || \quad || \quad \text{?} \quad || \quad || \quad || \quad || \quad || \quad || \quad || \quad \text{?} \quad || \quad || \quad || \quad || \quad \text{?} \quad || \quad || \quad || \\
 RS(K) : 28 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ \cancel{8} \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16 \ \cancel{17} \ 18 \ 19 \ 20 \ \cancel{21} \ 22 \ 23 \ \cancel{24} \ 25 \ 26 \ 27
 \end{array}$$

Hence $[26, 28] \cup [1, 9] \ni (K_9 = K_{12}) \in [10, 18] \ni (K_{18} = K_{21}) \in [19, 22] \ni (K_{22} = K_{25}) \in [23, 25]$; all the partitions share the same value, either 0 or 1. QED.

3.4.1.2 DES fixed points with weak keys

We adopt the following notations:

- For all $i \in [0, 16]$, we note $(L_i, R_i) \doteq LR_i$.
- The **plaintext** is $IP^{-1}(LR_0) = FP(LR_0)$ and the **ciphertext** is $FP(RL_{16})$.

Lemma 4 — *whenever K is a weak key*

$\forall i \in [0, 15], \forall j \in [1, 16], \quad LR_i = RL_j \Rightarrow LR_{i+1} = RL_{j-1}$.

Demonstration 3 (Proof of lemma 4)

$\forall i \in [0, 15], LR_{i+1} = (R_i, L_i \oplus f(R_i, \kappa))$, where $\kappa \doteq PC2(CD_i)$ does not depend on i , since all subkeys are equal. Consequently, $\forall j \in [1, 16], LR_{j-1} = (R_j \oplus f(L_j, \kappa), L_j)$. Thus:

$$\left\{ \begin{array}{lcl} RL_{j-1} & = & (L_j, R_j \oplus f(L_j, \kappa)) \\ & = & (R_i, L_i \oplus f(R_i, \kappa)) \\ & = & LR_{i+1}. \end{array} \right. \quad \begin{array}{l} \text{(Hypothesis)} \\ \text{QED} \end{array}$$

Definition 3 (DES fixed points with weak keys) A fixed point is a message M that **is not modified** by encryption (and thus decryption): $DES_K(M) = M$.

Property 2 DES has exactly 2^{32} fixed points when K is a **weak key**.

Demonstration 4 (Proof of property 2) Let $M = DES_K(M)$. It follows that $LR_0 = RL_{16}$, because IP is bijective. Hence, for all $k \in [0, 16]$, $LR_k = RL_{16-k}$ (Previous lemma 4 applied by induction.) In particular, $LR_8 = RL_8$, i.e. $L_8 = R_8$. Now, $\forall x \in \{0, 1\}^{32}$, there exists one and only one message $M(x)$ such that $LR_8 = (x, x)$, because $M \mapsto LR_8$ is bijection. Thus, the 2^{32} $M(x)$ are fixed point and there are no others. QED.

3.4.2 DES semi-weak keys and anti-fixed points

3.4.2.1 DES semi-weak keys

Definition 4 (DES semi-weak key) A couple of **semi-weak keys** (K, K') satisfies: $DES_K \circ DES_{K'} = Id$. If K is used to encrypt, then decrypting can be achieved by a second encryption with K' .

Property 3 (DES Semi-Weak Keys)

- **Weak keys** are **semi-weak keys**.
- There are **6 couples** of **strict semi-weak keys** (i.e. $K \neq K'$):

#	$PC1(K) = CD_0$	$PC1(K') = CD'_0$
1.	$\{01\}^{14} \times \{0\}^{28}$	$\{10\}^{14} \times \{0\}^{28}$
2.	$\{01\}^{14} \times \{1\}^{28}$	$\{10\}^{14} \times \{1\}^{28}$
3.	$\{01\}^{14} \times \{10\}^{14}$	$\{10\}^{14} \times \{01\}^{14}$
4.	$\{10\}^{14} \times \{10\}^{14}$	$\{01\}^{14} \times \{01\}^{14}$
5.	$\{0\}^{28} \times \{10\}^{14}$	$\{0\}^{28} \times \{01\}^{14}$
6.	$\{1\}^{28} \times \{10\}^{14}$	$\{1\}^{28} \times \{01\}^{14}$

Demonstration 5 (Proof of property 3) *If encryption with K followed by encryption with K' yields Id , then $\kappa_i = \kappa'_{17-i}$ for the 16 rounds $i \in [1, 16]$.*

- Thus: $\forall k \in \{1, 2, 4, 6, 8, 10, 12, 14, 15, 17, 19, 21, 23, 25, 27, 28\}$,

$$\boxed{PC2(LS^k(CD_0)) = PC2(RS^{k-1}(CD'_0))}.$$

- Moreover, as $RS = LS^{-1}$ and $LS^{28} = Id$,

$$\boxed{PC2(LS^k(CD_0)) = PC2(LS^{29-k}(CD'_0))}.$$

As already mentioned, $PC2$ does split into two halves. Hence a demonstration only on “C”.

- For $k = 1$,

$$\begin{array}{cccccccccccccccccccccccccccc} LS(K) : & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \cancel{10} & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & \cancel{19} & 20 & 21 & 22 & \cancel{23} & 24 & 25 & \cancel{26} & 27 & 28 & 1 \\ & || & || & || & || & || & || & || & || & \delta^? & || & || & || & || & || & || & || & \delta^? & || & || & || & \delta^? & || & || & \delta^? & || & || & || \\ Id(K') : & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \cancel{9} & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & \cancel{18} & 19 & 20 & 21 & \cancel{22} & 23 & 24 & \cancel{25} & 26 & 27 & 28 \end{array}$$

- Moreover, for $k = 28$,

$$\begin{array}{cccccccccccccccccccccccccccc} Id(K) : & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \cancel{9} & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & \cancel{18} & 19 & 20 & 21 & \cancel{22} & 23 & 24 & \cancel{25} & 26 & 27 & 28 \\ & || & || & || & || & || & || & || & || & \delta^? & || & || & || & || & || & || & || & \delta^? & || & || & || & \delta^? & || & || & \delta^? & || & || & || \\ LS(K') : & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \cancel{10} & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & \cancel{19} & 20 & 21 & 22 & \cancel{23} & 24 & 25 & \cancel{26} & 27 & 28 & 1 \end{array}$$

Thus, $\forall i \in [1, 28] \setminus [9, 18, 22, 25]$, $K_{i+1} = K'_i$ and $K_i = K'_{i+1}$.

- Furthermore, for $k = 2$,

$$\begin{array}{cccccccccccccccccccccccccccc} LS^2(K) : & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \cancel{11} & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & \cancel{20} & 21 & 22 & 23 & \cancel{24} & 25 & 26 & \cancel{27} & 28 & 1 & 2 \\ & || & || & || & || & || & || & || & || & \delta^? & || & || & || & || & || & || & || & \delta^? & || & || & || & \delta^? & || & || & \delta^? & || & || & || \\ RS(K') : & 28 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \cancel{8} & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \cancel{17} & 18 & 19 & 20 & \cancel{21} & 22 & 23 & \cancel{24} & 25 & 26 & 27 \end{array}$$

- And for $k = 27$,

$$\begin{array}{cccccccccccccccccccccccccccc} RS(K) : & 28 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \cancel{8} & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & \cancel{17} & 18 & 19 & 20 & \cancel{21} & 22 & 23 & \cancel{24} & 25 & 26 & 27 \\ & || & || & || & || & || & || & || & || & \delta^? & || & || & || & || & || & || & || & \delta^? & || & || & || & \delta^? & || & || & \delta^? & || & || & || \\ LS^2(K') : & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \cancel{11} & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & \cancel{20} & 21 & 22 & 23 & \cancel{24} & 25 & 26 & \cancel{27} & 28 & 1 & 2 \end{array}$$

Thus, $\forall i \in [1, 28] \setminus [9, 18, 22, 25]$, $K_{i+2} = K'_{i-1}$ and $K_{i-1} = K'_{i+2}$.

As a consequence, $\forall i \in [1, 28]$, $K_{i+1} = K'_i$. Indeed, $\forall i \in [9, 18, 22, 25]$, we have:

$$\begin{aligned} K_{i+1} &= K_{(i-1)+2} \\ &= K'_{(i-1)-1} & (k=2) \\ &= K_{(i-1)} & (k=1) \\ &= K'_{(i-1)+1} & (k=28) \end{aligned}$$

Idem, $\forall i \in [1, 28]$, $K_i = K'_{i+1}$. Hence, $\forall i \in [1, 28]$, $K_i = K_{i+2}$ and $K'_i = K'_{i+2}$. The relationships with other k will not bring additional information, since $\forall k, k - (29 - k)$ is odd. As for the “C” part, there are 4 semi-weak keys, depending whether $(K_1, K_2) = (0, 0), (0, 1), (1, 0)$ or $(1, 1)$.

Table 3.7: Register CD successive contents during a DES encryption with semi-weak key of either pair 3 or pair 4.

Round #	Register CD content
1	$\overline{CD_0}$
2 – 8	CD_0
9 – 15	$\overline{CD_0}$
16	CD_0

3.4.2.2 DES anti-fixed points with semi-weak keys

Definition 5 An *anti-fixed point* is a message M such that $DES_K(M) = \overline{M}$.

Property 4 (DES Anti-Fixed Points) DES has exactly 2^{32} anti-fixed points when using either semi-weak key from the couple 3 or 4.

The demonstration of the 2^{32} anti-fixed points requires the following assumption on the sub-keys. The key schedule is **anti-palindromic**, which means that: $\forall i \in [1, 16], CD_i = \overline{CD_{17-i}}$. This assumption holds only for the key couples 3 and 4, because if C_1 or D_1 is equal to either $\{0\}^{28}$ or $\{1\}^{28}$, then $C_{16} = C_1$ and $D_{16} = D_1$. For CD_{16} to be the logical invert of CD_1 , C_1 and D_1 must thus be equal to either $\{01\}^{14}$ or $\{10\}^{14}$. In this case, the register CD successive contents is given in Tab. 3.7. In this very table, CD_0 is $PC1(K)$; in encryption mode, the key is LS'ed once, hence the first inversion of the key. The keys of either type 3 or type 4 are assumed for the rest of the demonstrations.

Lemma 5 — whenever K is a semi-weak key of type 3 or 4
 $\forall k \in [0, 15], LR_k = \overline{RL_{16-k}} \Rightarrow LR_{(k+1)} = \overline{RL_{16-(k+1)}}$.

Demonstration 6 (Proof of the lemma 5)

- $\forall i \in [0, 15], LR_{i+1} = (R_i, L_i \oplus f(R_i, \kappa_{i+1}))$, where $\kappa_i \doteq PC2(CD_i) = PC2(CD_0)$ when $i \in [2, 8] \cup \{16\}$, the complemented otherwise (refer to Tab. 3.7.)
- Consequently, $\forall j \in [1, 16], LR_{j-1} = (R_j \oplus f(L_j, \kappa_j), L_j)$.
- Thus:

$$\left\{ \begin{array}{ll} RL_{15-k} &= (L_{16-k}, R_{16-k} \oplus f(L_{16-k}, \kappa_{16-k})) \\ &= (\overline{R_k}, \overline{L_k} \oplus f(\overline{R_k}, \overline{\kappa_{k+1}})) & \text{(Hypothesis + } \kappa_i \text{ property)} \\ &= (\overline{R_k}, \overline{L_k} \oplus f(R_k, \kappa_{k+1})) & (f(\overline{\cdot}, \overline{\cdot}) = f(\cdot, \cdot) \text{ for DES [71]}) \\ &= \overline{LR_{k+1}}. & QED \end{array} \right.$$

Demonstration 7 (Proof of property 4)

Let $\overline{M} = DES_K(M)$. Then, equivalently, $LR_0 = \overline{RL_{16}}$. Hence, for all $k \in [0, 16]$, $LR_k = \overline{RL_{16-k}}$ (Previous lemma 5 applied by induction.) In particular, $LR_8 = \overline{RL_8}$, i.e. $L_8 = \overline{R_8}$. Thus, $\forall x \in \{0, 1\}^{32}$, the message $M(x)$ such as $LR_8 = (x, \overline{x})$ is an anti-fixed point and there are no others. QED.

Table 3.8: Exhaustive list of weak and semi-weak DES keys.

#	Key	Dual	PC1(Key)	PC1(Dual)
1.	e001e001f101f101	01e001e001f101f1	55555550000000	aaaaaaa0000000
2.	fe1ffe1ffe0efe0e	1ffe1ffe0efe0efe	5555555fffffff	aaaaaaafffffff
3.	e01fe01ff10ef10e	1fe01fe00ef10ef1	5555555aaaaaaa	aaaaaaa5555555
4.	01fe01fe01fe01fe	fe01fe01fe01fe01	aaaaaaaaaaaaaa	55555555555555
5.	011f011f010e010e	1f011f010e010e01	0000000aaaaaaa	00000005555555
6.	e0fee0fef1fef1fe	fee0fee0fef1fef1	fffffffaaaaaaa	fffffff5555555
7.	0101010101010101	0101010101010101	00000000000000	00000000000000
8.	fefefefefefefefe	fefefefefefefefe	fffffffffffffff	fffffffffffffff
9.	e0e0e0e0f1f1f1f1	e0e0e0e0f1f1f1f1	fffffff0000000	fffffff0000000
10.	1f1f1f1f0e0e0e0e	1f1f1f1f0e0e0e0e	0000000fffffff	0000000fffffff

3.5 DES remarkable SCA properties

We recall that the studied architecture for the DES datapath is shown in Fig. 3.12. Compared with the versatile implementation shown in Fig. 3.6, the key schedule contributes to the critical path. The architecture is thus not optimal. However, as it is available in the first version of the SecMat ASIC (refer to Sec. A.1), it is used to realize the power attacks. The key schedule for this module is explicated in Tab. 3.6.

In this section, we use remarkable DES keys to exhibit power signatures in a view to grasping the nature of the information leakage via the power. DES weak (7 to 10) & semi-weak (1 to 6) keys were characterized in Sec. 3.4; They are also given in NIST/FIPS 74 [68], and reproduced in Tab. 3.8.

Two power traces (see Fig. 3.21) corresponding to two encryptions of the same message $\{0\}^{64}$ are acquired with:

1. the weak key 10, and
2. the semi-weak key 4.

Regarding their power dissipation, semi-weak keys can be partitioned into three classes:

1. semi-weak keys 1 & 2 toggle either none or all DFFs of C, resulting in 0 or 28 toggles,
2. semi-weak keys 3 & 4 toggle either none or all DFFs of CD, resulting in 0 or 56 toggles,
3. semi-weak keys 5 & 6 toggle either none or all DFFs of D, resulting in 0 or 28 toggles.

In a view to maximize the difference between the absence (LS^i , for i even) and the presence of toggles (LS^i , for i odd), the key was chosen amongst $\{3, 4\}$. With any weak key K , the key schedule does not consume any power, because the content of $PC1(K)$ is affected by neither LS^i , $i \in \mathbb{Z}$.

It is thus interesting to substract the trace acquired with the semi-weak key with the one acquired with a weak key, because the activity correlated with the message will be *partially* removed: only the activity linked to the key schedule will remain. Actually, to be accurate, the activity linked to the datapath LR would cancel only if averages over many messages for each key of the pair were substracted. As, in the present case, only one message is used, the difference is noisy.

It is plotted in Fig. 3.22. This figure shows that a lot — if not all — of information about the algorithm structure can be retrieved out of only two chosen traces. The rest of this section is devoted to the clarification of the link between the peaks in Fig. 3.22 and the datapath architecture.

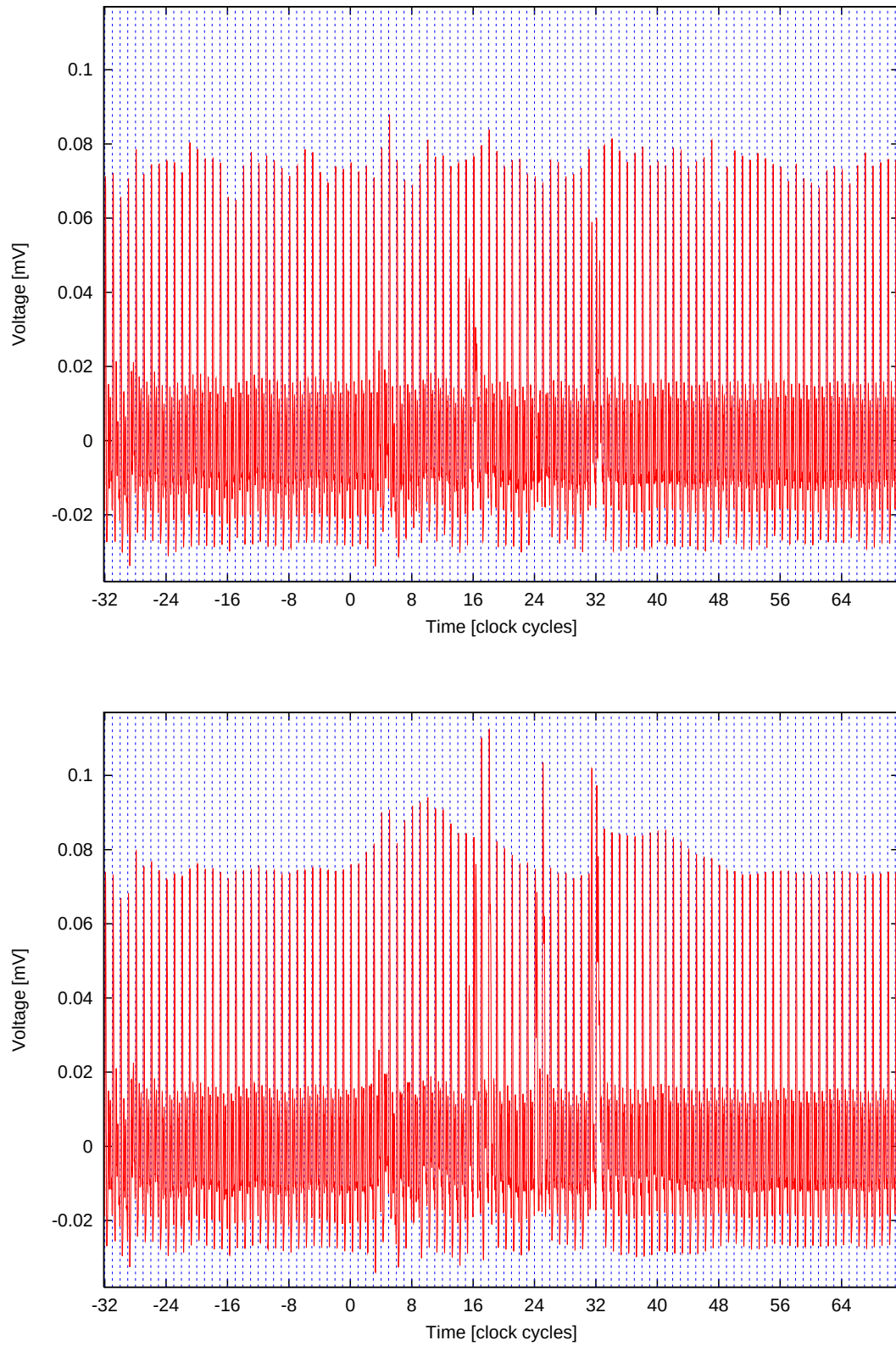


Figure 3.21: Encryption trace of the plaintext $\{0\}^{64}$ with the weak key $\{0\}^{64}$ (*upper.*)
Encryption trace of the same plaintext $\{0\}^{64}$ with the semi-weak key $\{00ff\}^4$ (*lower.*)

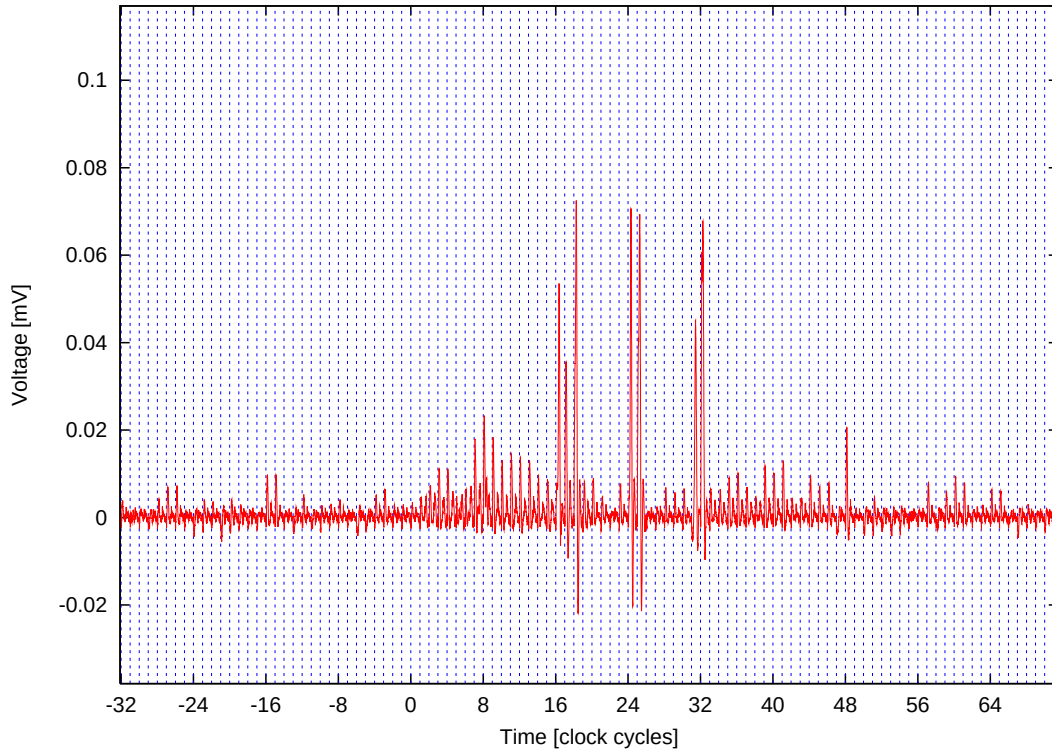


Figure 3.22: Difference of the two power traces plotted in Fig. 3.21 at constant scale.

3.5.1 Semi-weak keys

At the RTL level, the parity bits are not checked for. Therefore, the key “00ff00ff00ff00ff” can be injected into the datapath. The PC1 transformation will remove the (wrong) parity bits, resulting in a DES-correct semi-weak key equal to: “01fe01fe01fe01fe”.

As it can be seen from the schedule of Fig. 3.5, the theoretical power consumption caused by the key bits for one single encryption occurs at:

- Cycles [0-7], for the key loading into IF, and cycles [8-15] for the key flush out IF while the message overwrites it. This is depicted in Tab. 3.9. The XX value denotes data uncorrelated to the key. In cycles [0-7], it consists in the previous content of IF, for instance a cryptogram. In cycles [8-15], it is the message, differing from the key.
- Cycles [16-32], where the key is used as a master to derive the round keys. The content of the CD register is explicated in Tab. 3.10. The table was obtained from the data of the column “CD State” of Tab. 3.6 that presents the DES key schedule.

This journey through the DES pipeline is also represented in a snapshot, taken for the **Mentor Graphics** MODELSIM (*aka vsim*) event-driven simulator, in Fig. 3.23, with a testbench clock frequency of 100 MHz. The eight clock cycles required for the key and then the message loading, and the sixteen cycles of the encryption proper are delimited by cursors. The first cursor marks the clock cycle 0.

The number of bit toggles related to the key is given throughout the transit of the key. It is computed as follows: the number of bit flips caused by a key between two consecutive clock periods. For instance, when the key is loaded in IF (refer to Tab. 3.9):

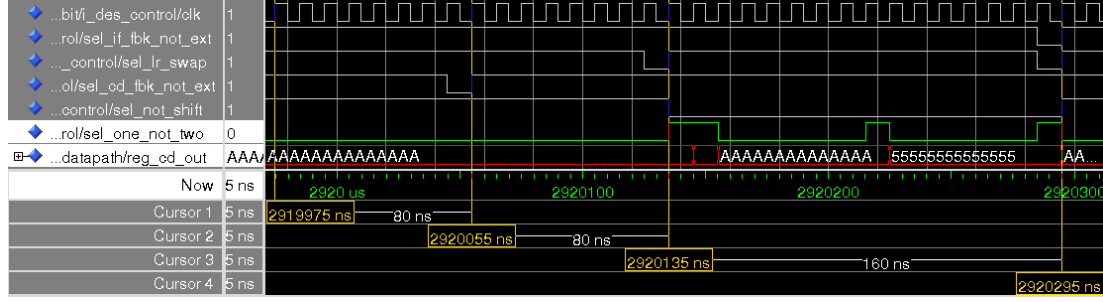


Figure 3.23: Transit of the key 00ff00ff00ff00ff in one DES encryption.

Table 3.9: Theoretical dissipation of the semi-weak key 00ff00ff00ff00ff in its transit in IF.

Time [clocks]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IF[1-8]	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
IF[9-16]	XX 00	ff 00	ff 00	ff 00	ff 00	ff 00	ff 00	ff 00	ff XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
IF[17-24]	XX XX	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
IF[25-32]	XX XX	XX 00	ff 00	ff 00	ff 00	ff 00	ff 00	ff 00	ff 00	ff XX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
IF[33-40]	XX XX	XX XX	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
IF[41-48]	XX XX	XX XX	XX XX	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	XX XX	XX XX	XX XX	XX XX	XX XX
IF[49-56]	XX XX	XX XX	XX XX	XX XX	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	XX XX	XX XX	XX XX	XX XX
IF[57-64]	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	00 ff	XX XX	XX XX	XX XX
Toggle count	0	8	16	24	32	40	48	56	56	48	40	32	24	16	8	0

- In cycle 0, the key first byte 0x00 replaces an arbitrary value (XX.) It is thus likely to cause $8/2 = 4$ bit flips. This is the average toggle rate, and is thus ignored.
- In cycle 1, the key first byte 0x00 causes the same transition as just mentioned. In addition, the key second byte 0xff replaces the value 0x00, which causes $|0x00 \oplus 0xff| = 8$ bit flips. Hence the contribution to the power consumption.

Although the same number 56 of antinomic register transfers occur at $t = 7$ and $t = 8$, the peak at $t = 8$ is higher than at $t = 7$. The fact that the byte IF[57-64] is more loaded than the others accounts for this difference. Indeed, the last byte of the register IF loads the RAM input, whereas the others merely load standard cells. This discrepancy is a first evidence of the dependency of the side-channel leakage with the net loads.

In the architecture used in SecMat (refer to the datapath shown in Fig. 3.12), the multiplexor that selects one shift type amongst the five $\{LS^i, i \in [-2, +2]\}$ (MUXPC2 in short) is situated

Table 3.10: Theoretical dissipation of the semi-weak key 00ff00ff00ff00ff in its transit in CD.

Time [clocks]	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
CD[1-56]	aa ⁷	55 ⁷	aa ⁷	aa ⁷	aa ⁷	aa ⁷	aa ⁷	aa ⁷	aa ⁷	55 ⁷	55 ⁷	55 ⁷	55 ⁷	55 ⁷	55 ⁷	55 ⁷	aa ⁷
Toggle count	0	56	56	0	0	0	0	0	0	56	0	0	0	0	0	0	56

between the register CD and the PC2 function. The subkey used in one round is thus sampled in CD at the beginning of the next round. The selections operated by MUXPC2 thus anticipate the CD register transfers. In the current encryption scenario, MUXPC2 only chooses between LS^1 and LS^2 , under the control of a signal named `sel_one_not_two`. This signal builds up the key schedule; its variations are represented in green in Fig. 3.23. When `sel_one_not_two` changes values, the lines starting from MUXPC2 and leading to CD and the PC2 are charged. The values those lines convey are limited to aa^7 and 55^7 . Thus, each time `sel_one_not_two` toggles, $|aa^7 \oplus 55^7| = 56$ lines also toggle in response. This combinatorial power consumption adds up to the one caused by sequential parts of the datapath (register transfers.) As the signal `sel_one_not_two` is computed from a finite state machine, it takes its new value after a small amount of time, leading to peaks slightly late w.r.t. the clock rising edge. In addition, this signal may glitch, *i.e.* not take its value at once. In the case of an incomplete transition, the multiplexors are likely to filter out the spurious event. However, if the transition was complete (such as: $0 \rightarrow 1 \rightarrow 0$ in one clock period), then the combinatorial power consumption of the lines charged by MUXPC2 is roughly doubled. It is exactly doubled if the glitch duration was large enough to let all the lines have in turn full transitions. Let apart the glitch complications, the combinatorial transitions just described occur at clock periods 16, 18, 24, 25, 31 and 32. This phenomenon can be seen in the power traces. For instance, Fig. 3.24 shows a magnified view of Fig. 3.22, where it clearly appears that:

- **at clock 16**, there is only a combinatorial dissipation (`sel_one_not_two`↑),
- **at clock 17**, there is only a sequential dissipation (CD: $aa^7 \rightarrow 55^7$) and that
- **at clock 18**, both types of dissipation add up (CD: $55^7 \rightarrow aa^7$ + `sel_one_not_two`↓), with a few nanoseconds delay for the combinatorial contribution.

The overall theoretical power model for the semi-weak key studied in this section is plotted in Fig. 3.25. The use of the null weak key would lead to an “all zero” dissipation profile. Indeed, neither the loading in IF nor in CD trigger dissipation, since $|value \oplus value| = 0$. The graph shown in Fig. 3.25 also represents the difference of the bit toggles for the semi-weak and the absence of bit toggles for the weak key. The hardware activated by the key bits is highlighted in Fig. 3.26. It is thus an approximation of the real traces difference computed in Fig. 3.22. Notice nonetheless that the number of bit flips is only a first order estimate for the power physical dissipation. A more accurate description would weight the bit flips with a quantity related to the bit capacitive load. Within this model, it becomes relevant to add up dissipations. In the Fig. 3.22, the additions are only illustrative, because the operands are physically incommensurable.

3.5.2 Weak keys

The peak occurring sixteen clock periods after the end of the encryption (*i.e.* at period 48) is caused by a default of dissipation for the null-key trace. With the null-key, not only encryption is equal to decryption, but also the subkeys are all equal to zero. In the studied DES architecture, the LR register does not have an enable (power saving strategies can be achieved at a higher level, using for instance clock gating.) Thus, another encryption starts afresh right after the requested one. But as encryption is tantamount to decryption, the ciphertext is actually decrypted during rounds 17 to 32.

As shown in Tab. 3.11, the plaintext is retrieved again sixteen periods after the ciphertext has been output. We refer to this phenomenon as a **resurgence**. Given that the analyzed operation processes one single block of data, all the control signals are only activated for one encryption. As already indicated, this does not impact the key schedule; however, it does impact the datapath. The DES algorithm indeed demands that in the last round, the two message halves must not be swapped. As indicated in Tab. 3.11, the signal that controls the

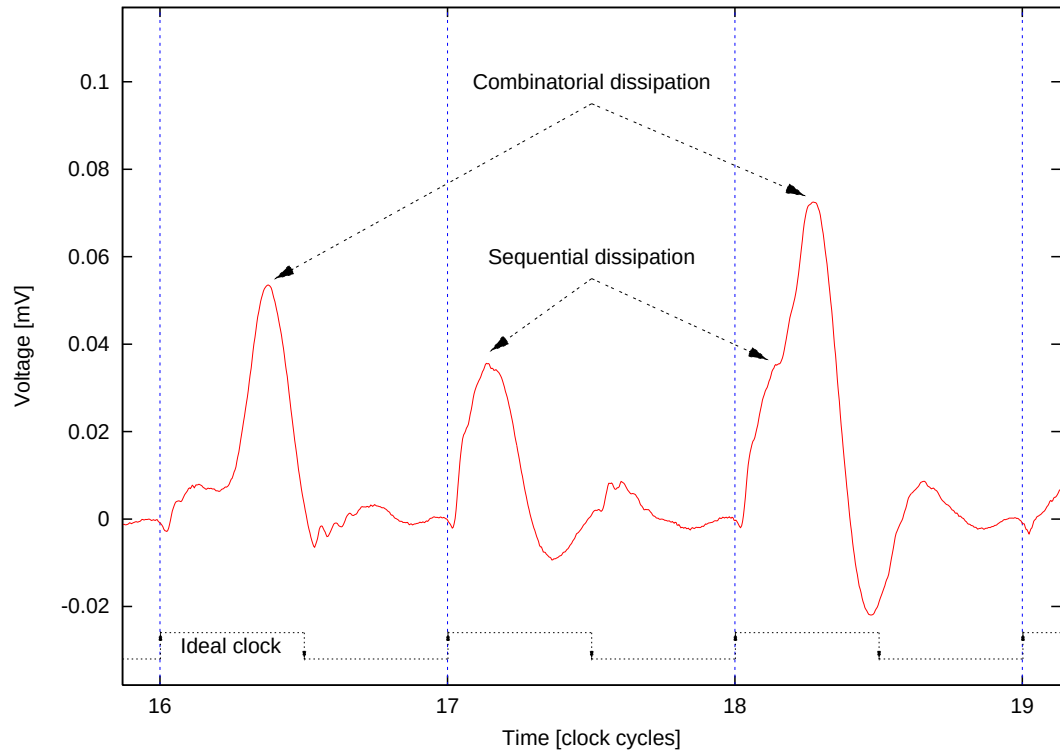


Figure 3.24: Zoom on the clock periods 16, 17 and 18 of the difference of the power traces plotted in Fig. 3.22. The clock frequency is 66 MHz, hence its period is about 15 ns.

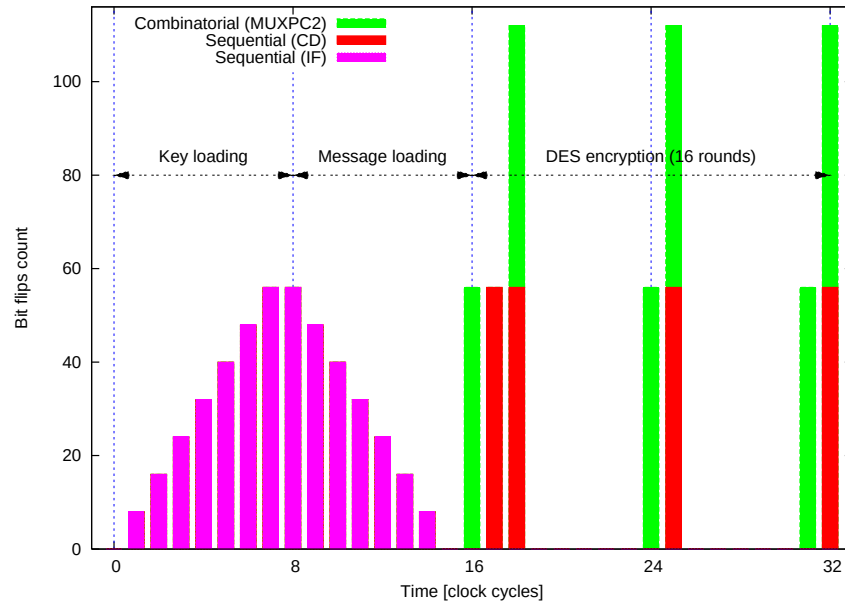


Figure 3.25: Theoretical dissipation related to the key transit when it is semi-weak.

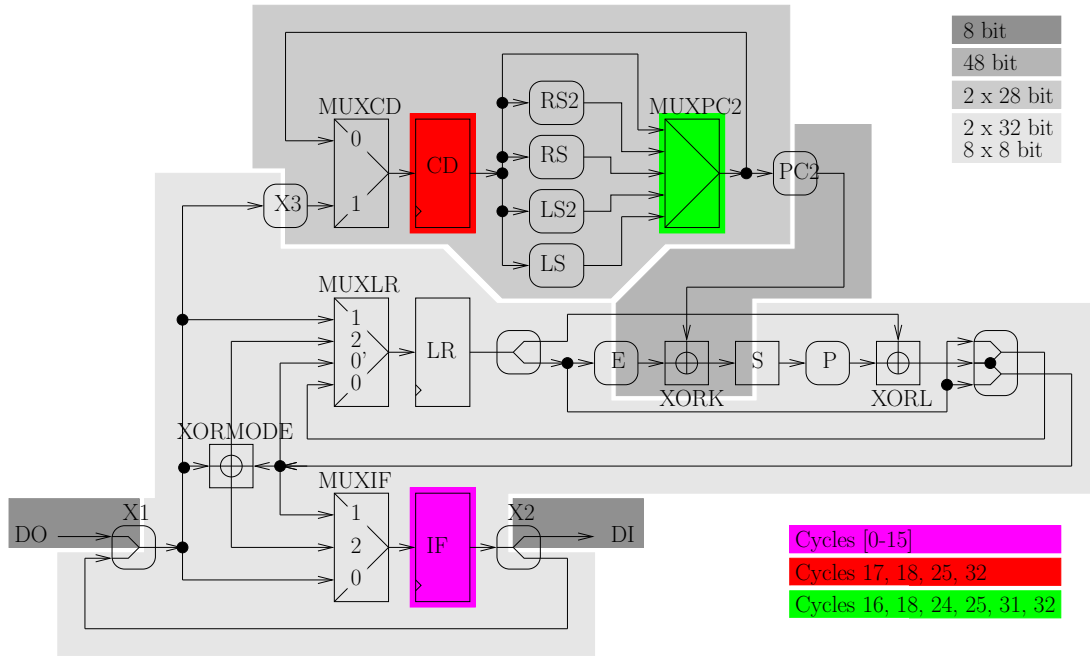


Figure 3.26: Pieces of hardware activated in the DES datapath logic by a semi-weak key transit. Contrast this figure with Fig. 3.12 at page 66.

Table 3.11: Register LR contents when using a weak key, a symmetrical message, and a single block DES-ECB encryption.

Round #	0	$i \in [1, 15]$	16	$i \in [17, 31]$	32	$i \in [33, 47]$	48	...
Clock	16	[17, 31]	32	[33, 47]	48	[49, 63]	64	...
LR contents	LR_0	LR_i	RL_{16}	RL_{32-i}	$RL_0 = LR_0$	LR_{i-32}	$LR_{16} \neq RL_{16}$	XX
Swap L $\leftrightarrow$ R?	yes	yes	no	yes	yes	yes	yes	yes

swap $L \leftrightarrow R$ is inactive at the end of the nominal encryption, at round 16, but remains active afterwards. The reason why the plaintext is recovered at round 32 is that it is specific: it is symmetrical, meaning that it verifies $RL_0 = LR_0$. This also explains why there is only one resurgence at round 32. In fact, there would be an infinite number of resurgences if the LR register was swapped every sixteen rounds. Because this is no longer the case after the single block ECB-mode encryption, the ciphertext obtained at clock period $32 + 16 = 48$ differs from the actual ciphertext LR_{16} . Consequently, the decryption is, this time, not consistent with respect to the plaintext.

Now, as indicated in Tab. 3.11, one clock period before the resurgence, the LR register contains the value RL_1 . At the resurgence, the transition $RL_1 \rightarrow LR_0$ occurs. As for all i , $L_{i+1} = R_i$, there is no change in R: $|L_1 \oplus R_0| = 0$. This lack of power consumption does not happen for the semi-weak or regular keys, since the hypothesis of key-schedule independence is not true, which does not allow for any resurgences. Hence the **positive peak** showing up at clock period period 48 in Fig. 3.22.

3.5.3 SCA properties generalization for arbitrary keys

The power signatures just illustrated in Sec. 3.5.1 and 3.5.2 can in fact be put forward with arbitrary keys, albeit using much more than two traces. The results previously obtained with one sole weak key and another semi-weak key was essentially pedagogical.

When random keys and messages traces are available through a side-channel, the following selection function can be used:

$$\underbrace{\text{IF}}_{\text{Initial state}} \oplus \underbrace{\text{X1}(\text{X2}(\text{IF})[1, 56] \parallel \text{DI})}_{\text{Final state}}, \quad (3.2)$$

where X1 and X2 have been defined in Fig. 3.14. The content of the register IF can be either “IP(key)” at clock 8 or “IP(msg)” at clock 16. The IF register is always fed with fresh data from the RAM; the sole exception to this behavior happens at clock 31, when IF samples the last round output in parallel. The 8-bit input from the RAM, term denoted “DI” in Eqn. (3.2), is chosen equal to:

- **at clock 8**: the first byte of the message, because in the architecture of Fig. 3.12, the message is loaded right after the key.
- **at clock 16**: the memory contents at address 0x00, because this is the default address selected by the DES datapath controller.

The Eqn. (3.2) basically expresses the fact that, in IF, the new incoming byte “DI” flushes the previously read byte. The last byte $\text{IF}[57, 64]$ is available at the RAM input as DO.

Differential traces are computed with the selection function (3.2) from the accumulation of 20 000 power traces using random keys and messages, acquired at 32 MHz. The two graphs represented in Fig. 3.27 shows that the key and the message remain consistent during their loading and unloading from the register IF. The two signatures are analog for the key and the message. They are nonetheless offset by eight clock periods (refer to Fig. 3.5), because key loading and unloading in IF occur at clock cycles $[0, 8[\cup [8, 16[$ (as in Fig. 3.22), whereas message loading and unloading in IF occur at clock cycles $[8, 16[\cup [16, 24[$. Despite the data traveling through IF is different (the key, followed by the message), the same hardware (register IF) is exercised. The loading and unloading signature for the key is equally intense to the one for the message because parity bits are not stripped off yet: in both cases, all the bits contribute to the dissipation signature.

Table 3.12: Maximum peaks amplitude at selected clock cycles for two differential traces.

Clock cycle	# 8	# 16
Traces difference (Fig. 3.22 at page 79)	23.31 mV	53.52 mV
Differential trace (Fig. 3.27(upper) at page 86)	17.70 mV	24.97 mV

3.5.3.1 Key signature in SecMat V1 datapath

On the one hand, the key propagation continues to sign, because:

$$LS \circ PC1 \approx PC1 \circ CLS_8, \quad (3.3)$$

where CLS denotes the “Circular Left Shift” operation. In fact, the equation (3.3) holds on the restriction $[1, 56] \setminus \{8, 16, 24, 28\} \cup \{36, 44, 52, 56\}$, hence for $6/7$ of register CD bits. The key loading/unloading corresponds to the transfers $K \leftarrow CLS_8(K)$ in IF (also refer to Eqn. (3.2)), because the input/output of every material is done on a byte basis. Consequently:

$$\begin{aligned}
& |K \oplus CLS_8(K)| && // [1] \text{ Matches selection function (3.2) for } 7/8 \text{ of the bits} \\
\approx & |PC1(K \oplus CLS_8(K))| && // [2] \text{ True for the entire key but the parity bits} \\
= & |PC1(K) \oplus PC1 \circ CLS_8(K)| && // [3] \text{ Linearity of PC1} \\
\approx & |PC1(K) \oplus LS \circ PC1(K)|. && // [4] \text{ Observation for } 6/7 \text{ of the bits made in (3.3)} \quad (3.4)
\end{aligned}$$

The equation (3.4) happens to concern the key without the following bits:

- $\{8 \times i, i \in [1, 8]\}$ (because of [1]),
- $\{8 \times i - 7, i \in [1, 8]\}$ (because of [2]),
- $\{PC1^{-1}(i), i \in \{8, 16, 24, 28\} \cup \{36, 44, 52, 56\}\} = \{1, 2, 3, 36, 7, 6, 5, 4\}$ (because of [4].)

In short, there are 23 key bits that are left apart (since bit 1 is shared by sets [2] and [4]), including the eight parity bits. Finally, the result is that for $\frac{64-23}{64} \approx 64\%$ of the bits, the selection function (3.2) matches too a register transfer in CD for the first left shift. This ratio can be verified on the experimental traces. The table 3.12 shows the maximum peak amplitude for the traces difference and the differential traces selection using Eqn. (3.2). As the trace come from two different acquisitions, the amplitudes cannot be compared in absolute value, but only relatively to a reference. The reference peak is chosen equal to the dissipation occurring in register CD at clock cycle #8. The peak at clock #16 serves as an evaluator of the correlated activity in the register CD. Compared to that of the difference, the activity is $\frac{24.97/17.70}{53.52/23.31} \approx 61\%$, which is very similar to the expected value 64 %. The small discrepancy can be explained by variations amongst the bits of register CD regarding their dissipation, by the noise in Fig. 3.22 due to the use of only two curves, and by the fact that the peak maximum is not the most appropriate metric.

Relations similar to (3.4) hold whenever there is a single left shift in the key schedule. As reminded in Tab. 3.6, this occurs at rounds 1, 2, 9 and 16. Notice that in the SecMat V1 architecture (*cf* Fig. 3.12), the correct number of key shifts is selected during the round, and the key is latched in register CD at the next rising edge of the clock. This explains why sequential peaks are visible at rounds 2, 3, 10 and 17. A more accurate analysis of rounds [1,16] is done in Tab. 3.13. But for sure, the differential trace of Fig. 3.27(upper) is an improved approximation (from a statistical viewpoint) of Fig. 3.22.

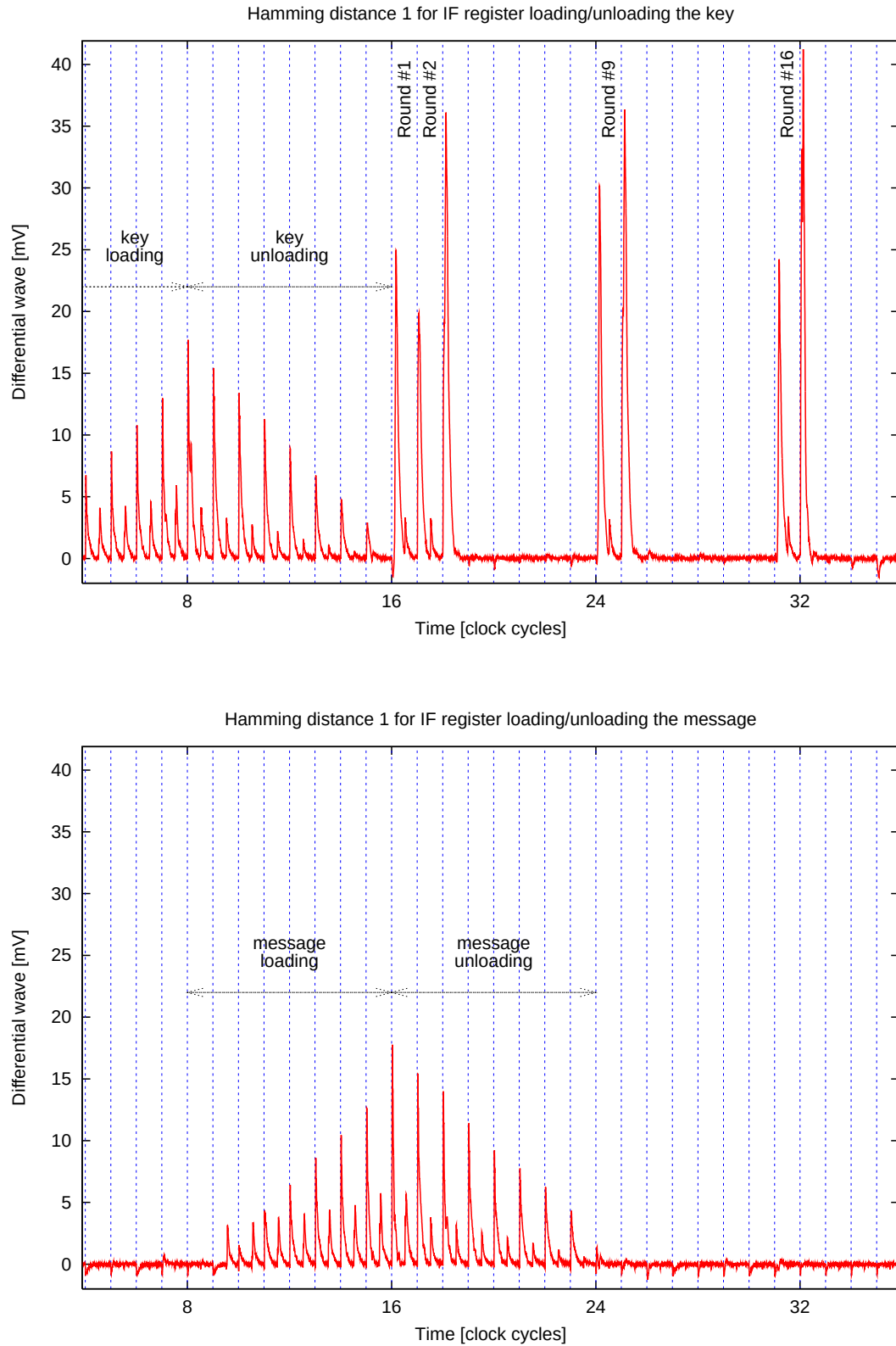


Figure 3.27: Signature of key loading / unloading in IF, plus the CD activity (*upper.*) Differential trace using the analog selection function (3.2), but operating on IP(msg) instead of IP(key) (*lower.*)

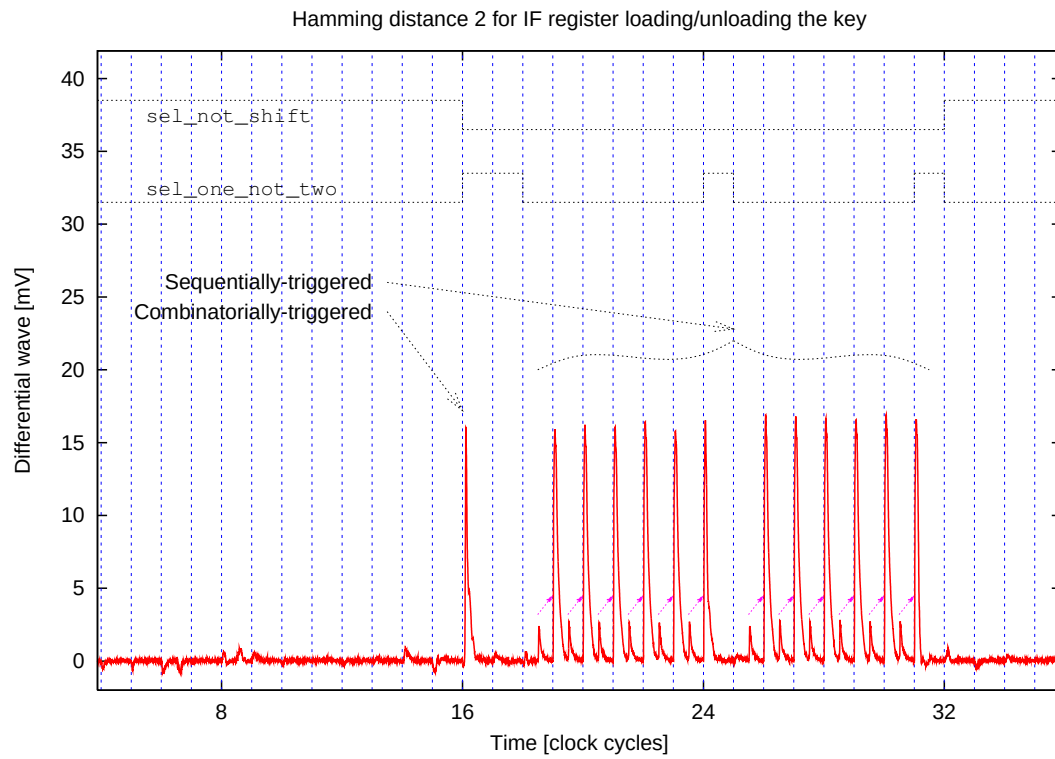


Figure 3.28: Signature of distance-2 key loading / unloading in IF and of LS^2 in CD (3.5).

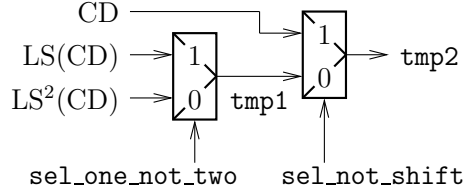


Figure 3.29: Part of the $5 \rightarrow 1$ MUXPC2 (refer to Fig. 3.12) activated during an encryption.

Incidentally, the Eqn. (3.3) can be extended to LS^2 :

$$LS^2 \circ PC1 \approx PC1 \circ CLS_{16}. \quad (3.5)$$

There is an equality on the set $[1, 56] \setminus \{9, 1, 10, 2, 11, 3, 44, 36\} \cup \{15, 7, 14, 6, 13, 5, 12, 4\}$, hence for $40/56$ bits of CD. This property can be shown in traces, by considering the selection function presented in Eqn. (3.2), but composed by itself. The result is shown in Fig. 3.28. The amplitude of the double left shift (*i.e.* LS^2) peaks is consistent with that of the single: $\frac{16.19 \text{ mV}}{19.88 \text{ mV}} \bigg/ \frac{56 - 16}{56 - 8} \approx 1$.

The distinction between sequentially-triggered and combinatorially-triggered peaks has already been observed in Sec. 3.5.1. A finer analysis leads to the explanation of the repartition of sequentially-triggered and combinatorially-triggered peaks between Fig. 3.27 and 3.28. The hardware that is concerned is the register CD and the multiplexer MUXPC2. Only the relevant part of the latter is analyzed; its internal 56-bit buses `tmp1` and `tmp2` are defined in Fig. 3.29. The detailed chronology of the events transmitted by the control to the datapath is provided in Tab. 3.13. In this table, “ $1/2$ ”, “`shift`” and “`clk`” are short names for the control signals “`sel_not_shift`”, “`sel_one_not_two`” and “`clock`”. The activation of a combinatorial dissipation on the Hamming distance 2 expressed by Eqn. (3.5) within clock period 16 is due to the conjunction of two facts:

1. `sel_one_not_two` = 0 in the idle state (it is a mere contingency of the control behavior) and
2. `sel_not_shift` is faster than `sel_one_not_two`, basically because in order to compute `sel_one_not_two`, the control combinatorial logic must evaluate two Boolean conditions: (1) is the DES co-processor in encryption mode? and (2) how many shifts are there in the current round? The first Boolean is indeed `sel_not_shift`. Consequently, the evaluation of `sel_one_not_two` traverses a deeper control logic than the evaluation of `sel_not_shift`.

Thus, the bus `tmp2` has a first transition: $CD_0 \rightarrow LS^2(CD_0)$. Then, `sel_one_not_two` changes from 0 to 1, which causes a second transition on `tmp2`: $LS^2(CD_0) \rightarrow LS(CD_0)$. The second transition is selected by Eqn. (3.3), because this selection function is not **oriented in time**: the signature of “initial $\rightarrow$ final” is identical to that of “final $\rightarrow$ initial”. In our example, $|LS^2(CD_0) \oplus LS(CD_0)| = |LS(CD_0) \oplus LS^2(CD_0)|$. The first “combinatorial” transition in clock #16 (*cf.* Fig. 3.27) is larger than the second one (*cf.* Fig. 3.28), because:

- when `sel_one_not_two` toggles, the two multiplexers toggle, whereas
- when `sel_one_not_two` toggles, only the selected one toggles.

The idea is that upstream changes in the datapath propagate downstream. Thus a change is all the more dissipative as it happens high in the datapath traversing multiplexers.

Table 3.13: Propagation of data through MUXPC2 (refer to Fig. 3.29) during the first three cycles of encryption.

Event	clk↑ (16)	$\overline{\text{shift}}\downarrow$	$1/\overline{2}\uparrow$	clk↑ (17)	clk↑ (18)	$1/\overline{2}\downarrow$	clk↑ (19)
CD ^a	CD ₀	CD ₀	CD ₀	CD ₁	CD ₂	CD ₂	CD ₃
tmp1	LS ² (CD ₀)	LS ² (CD ₀)	LS(CD ₀)	LS(CD ₁)	LS(CD ₂)	LS ² (CD ₂)	LS ² (CD ₃)
tmp2	CD ₀	LS ² (CD ₀)	LS(CD ₀)	LS ² (CD ₁)	LS(CD ₂)	LS ² (CD ₂)	LS ² (CD ₃)
Transition	∅	Eqn. (3.5)	Eqn. (3.3)	Eqn. (3.3)	Eqn. (3.3)	Eqn. (3.3)	Eqn. (3.5)

^aNotice that the register CD is updated synchronously with every rising edge of `clk`; by definition: $CD_1 \doteq LS(CD_0)$, $CD_2 \doteq LS^2(CD_1)$, $CD_3 \doteq LS^2(CD_2)$, *etc.* (also refer to [71] and to Tab. 3.6 at page 72.)

In the rest of the encryption (not detailed in Tab. 3.13 to keep it readable), the internal buses `tmp1` and `tmp2` only have $LS^1(CD_i) \rightarrow LS^2(CD_i)$ transitions (or *vice-versa*.) As a result, the Fig. 3.28 does not have any additional combinatorial peaks. The sequential peaks correspond to $CD_i \rightarrow CD_{i+1}$ transitions in CD (for $i \in [0, 16]$.) The sequence of `sel_not_shift` and `sel_one_not_two` (without delays) is superimposed to Fig. 3.28 in order to help differentiate the double from the single left shifts while the key schedule unrolls.

It can be noticed that at the low frequency of the acquisition, the key schedule is completed well before the half of the clock period. In the front latch of the CD register DFFs will thus sample in advance the new value CD_{i+1} at the falling edge of the clock. This has the effect of **announcing** a future sequential dissipation in the register. The power signatures of these announcements is reported in Fig. 3.28 with small arrows.

3.5.3.2 Message signature in SecMat V1 datapath

On the other hand, the message is mutated in such a way that there remains almost no correlation between the plaintext and the first round contents of LR. Thus, during the encryption proper, the Fig. 3.27 does not disclose any correlation peaks.

Peaks corresponding to the message transformation during the encryption can be produced by *ad hoc* selection functions, presented in Sec. 3.6.

3.5.3.3 Conclusion on DES remarkable SCA properties

The properties shown in this section can be used to reverse-engineer the architecture of the DES crypto-processor. For instance, the differential traces plotted in Fig. 3.27 clearly show that eight clock cycles are required to load the key, then the message, and that the encryption is iterative. Thus, if the implementation is secret, then power analysis is a means to discover it.

From a practical point of view, we used the semi-weak key $\{00\text{ff}\}$ ⁴ to locate the encryption rounds on the oscilloscope screen. The signature shown in Fig. 3.21 made it possible to configure the horizontal range in such a way only the relevant clock cycles were acquired.

To summarize this section on DES side-channel properties, the study of signature given by a trivial differential power analysis has taught us that:

- Both register transfers and combinatorial logic (especially dataflow “conditionals”, such as multiplexors.) can induce vulnerabilities, because they resynchronize the data.
- The load of the nets, but also the amount of downstream logic, impact the differential signature.

- Signatures follow the data throughout its journey in the datapath. This remark is valid only for the data that are not altered during the computation. This is of course true for DES keys but not for the messages, as recalled in Fig. 3.27.

3.6 Explanation for the differential traces using HW and HD selection functions

The differential traces obtained in Sec. 2.4.2 and plotted in Fig. 2.21 were expected to present a peak at the dates indicated by a vertical bar. It has been observed that other peaks appeared. Their presence is accounted for in this section.

The acquisition campaign is tailored for a DPA attack: the key is constant, whereas the messages change. It must be noted that the additional peaks observed in the differential traces are not a consequence of the key being constant. Similar differential traces are indeed obtained from traces where both keys and messages vary.

3.6.1 Interpretation of the differential trace using HW

Let us define the following selection functions collection:

$$\forall i \in [0, 15], \quad |f(R_i, K_{i+1})|, \quad (\text{Output of Sboxes at round } \#i + 1) \quad (3.6)$$

where f is the Feistel function for DES, as defined in Fig. 3.19. Notice that the Hamming weight of every weighting function in Eqn. (3.6) satisfies $|f(R_i, K_{i+1})| = |L_i \oplus R_{i+1}| = |L_i \oplus L_{i+2}|$. Some differential traces, using the weighting functions from Eqn (3.6), are represented in Fig. 3.30.

The weak and spread peak observed in the interesting clock period is followed by another peak, occurring slightly after the next clock rising edge. This peak is not due to a register transfer, otherwise its slope would be greater and it would coincide exactly with the clock rising edge.

The origin of the peak happens to be a “glitch” for $i \neq 15$: the differential traces for $i < 15$ are alike, and discussed below. The triple signature at $i = 15$ is due to a singular computation ending. A detailed analysis of the phenomenon is given later on in Sec. 3.6.2.3.

Glitches correlated to the round message occur at the output of XORK and XORL gates. The nature of the glitches is discussed below:

- The PC2 input of XORK is valid late in the clock period because of the MUXPC2 traversal. Thus the input from register LR arrives first, causing a glitch at the output of XORK.
- The outputs of the sboxes take awhile to be computed. Consequently, the L input of the XORL gate is ready well before S, thus causing a glitch at the output of XORL.

However, those two glitches yield respectively the following activity at the XOR gates output:

$$R_{i+1} \oplus R_i, \quad (\text{for gate XORK}) \quad (3.7)$$

$$L_{i+1} \oplus L_i. \quad (\text{for gate XORL}) \quad (3.8)$$

The origin of the observed peak is thus different. Actually, the XOR gates are dissymmetric w.r.t glitches. Whatever the steady input, a glitch at the other input does propagate. However, the power signature depends on the **value** of the steady input. This fact is illustrated in Tab. 3.14, obtained from electrical simulations on an exclusive-or standard cell extracted model.

In CMOS logic, an exclusive-or Boolean function cannot be realized with only one layer of transistors. In fact, a CMOS gate g satisfies $g(0, \dots, 0) = 1$ and $g(1, \dots, 1) = 0$. The first equality is obviously violated if $g = \text{XOR}$. As a consequence, some inputs must be inverted before being used. The exclusive-or gate is made up of two transistor layers, which explains

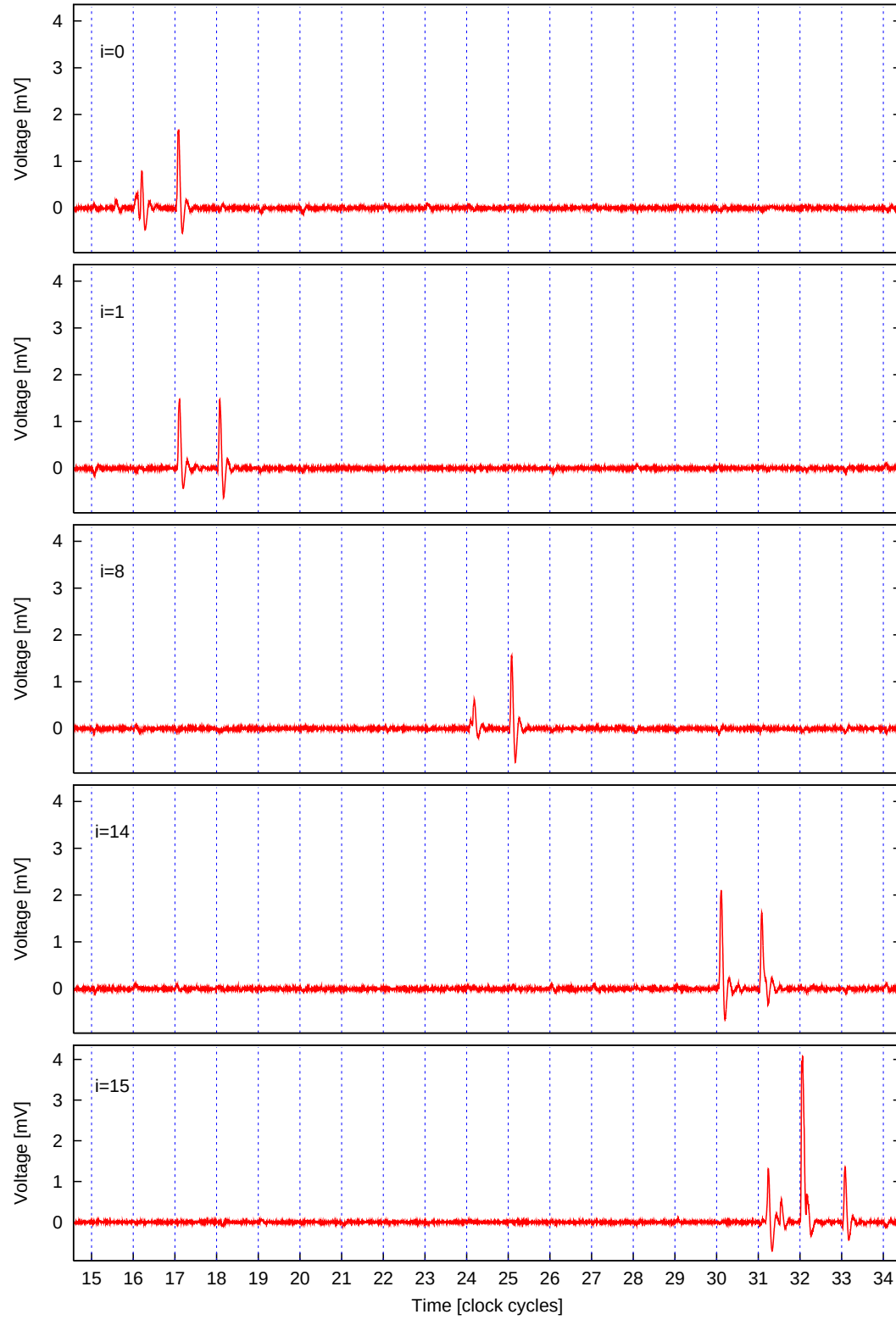


Figure 3.30: Differential traces weighted with the Hamming weight of Eqn. (3.6) for $i = 0, 1, 8, 14$ and 15 .

the two peaks in the power signatures shown in Tab. 3.14. The instant power can be negative, when the gate restitutes charges loaded in parasitic capacitances to the power supply. But after a complete toggle, the gate has globally spent energy: the integral of the power consumed by the gate is positive. The average energy dissipated by an event propagation from input b to the output z is:

- 9.46 pJ when the steady input a equals 0, but
- 11.87 pJ when the steady input a equals 1.

A dissymmetry can also be observed when the roles of a and b are permuted. Similar observations can be made for other logic gates. But the relevance of the dissymmetry for the XOR gate is that its restriction to one input is reversible: if the steady input is a and the output is z , then the second input b is known: $b = a \oplus z$. As a consequence, whatever the activity of input b (even unknown), the dissymmetry is correlated to a .

The XOR dissymmetry w.r.t. glitches yields the following power signatures:

$$\text{PC2}(\text{CD}_i), \quad (\text{for gate XORK}) \quad (3.9)$$

$$\text{R}_{i+1} \oplus \text{L}_i. \quad (\text{for gate XORL}) \quad (3.10)$$

The second peak observed in Fig. 3.30 ($i < 15$) results from the dissymmetry of XORL, that results in the bias of Eqn. (3.10). This peak is larger than the one generated by the substitution boxes during the previous round because the XOR gates glitch is synchronized (indirectly) with the clock; the clock has a rising edge, which causes the DFFs of register L to evaluate, which in “second hand” transmit the variation across XORL. The shape of the peak is thus smooth, because of the slight propagation time variations in L DFFs. The peak is not totally identical for all $i \in [0, 14]$ because the would-be steady input can actually evaluate fast: it was indeed shown in Fig. 3.18(a) that the sboxes outputs can be valid very early. It is probable that many DPAs realized thanks to an Hamming weight selection function and reported as successful in the literature actually exploited a glitch instead of the intended substitution boxes outputs.

3.6.2 Interpretation of the differential trace using HD

Following the analytical approach of Sec. 3.6.1, the following function family is defined:

$$\forall i \in [0, 15], \quad \text{LR}_i \oplus \text{LR}_{i+1}. \quad (\text{Register transfer at round } \#i + 1) \quad (3.11)$$

Some of them are plotted in Fig. 3.31. The plots are similar for $i = 2$ to 13. The plots for $i = 0, 1$ and $i = 14, 15$ have side-effects.

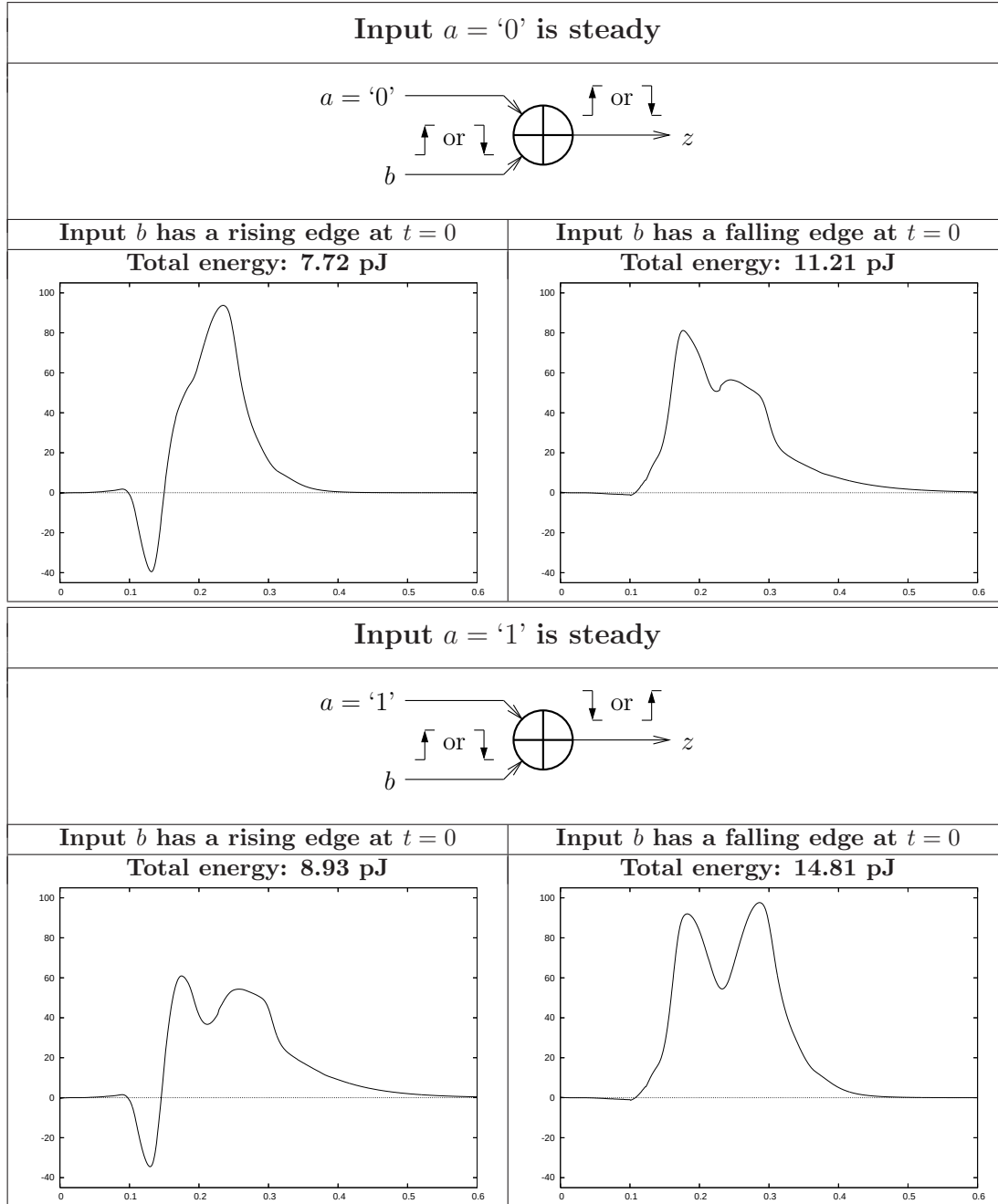
3.6.2.1 Interpretation of HD at rounds [3,14]

The differential curves for $i \in [2, 13]$ feature three peaks, occurring just after three consecutive clock edges. These curves share the property that the sum of the two outer peaks yield the central peak. This amazing property is shown in Fig. 3.32(c) on the example of $i = 8$.

The reason for this conservation property lies in the twisted ladder structure of DES. For $i \in [1, 14]$,

$$\left\{ \begin{array}{lcl} \text{LR}_{i-1} \oplus \text{LR}_i & = & (\text{L}_{i-1} \oplus \text{L}_i) \parallel (\text{R}_{i-1} \oplus \text{R}_i) \\ \text{LR}_i \oplus \text{LR}_{i+1} & = & (\text{R}_{i-1} \oplus \text{R}_i) \parallel (\text{L}_{i+1} \oplus \text{L}_{i+2}) \\ \text{LR}_{i+1} \oplus \text{LR}_{i+2} & = & (\text{L}_{i+1} \oplus \text{L}_{i+2}) \parallel (\text{R}_{i+1} \oplus \text{R}_{i+2}). \end{array} \right. \quad // \text{ See the note below}$$

Table 3.14: Instant power and total energy consumed by various glitches on an XOR gate. The graphs' time unit is [ns] and the power unit is [μW].



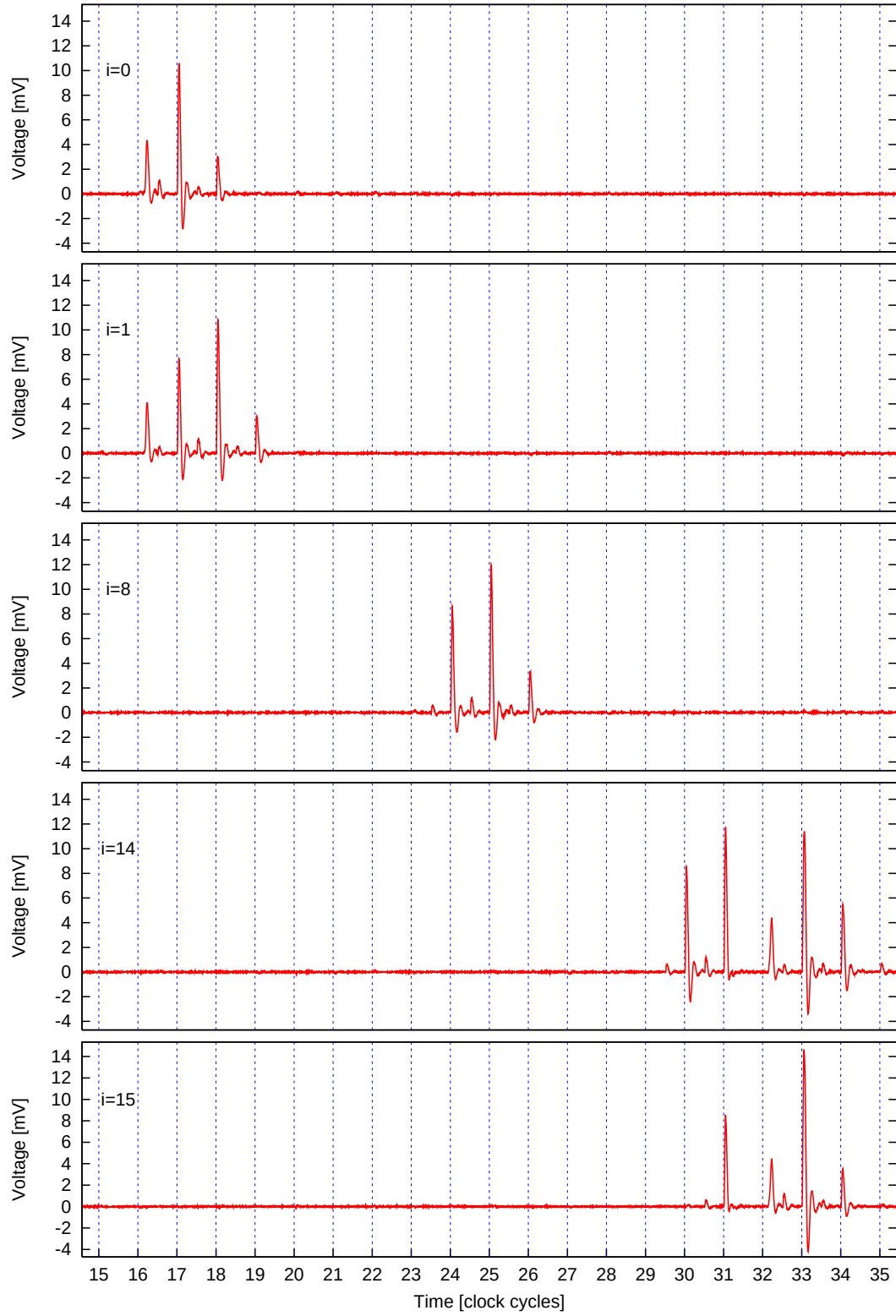


Figure 3.31: Differential traces weighted with Eqn. (3.11) for $i = 0, 1, 8, 14$ and 15 . The traces corresponding to $i \in [2, 13]$ are not plotted because they are all similar (to that of $i = 8$.)

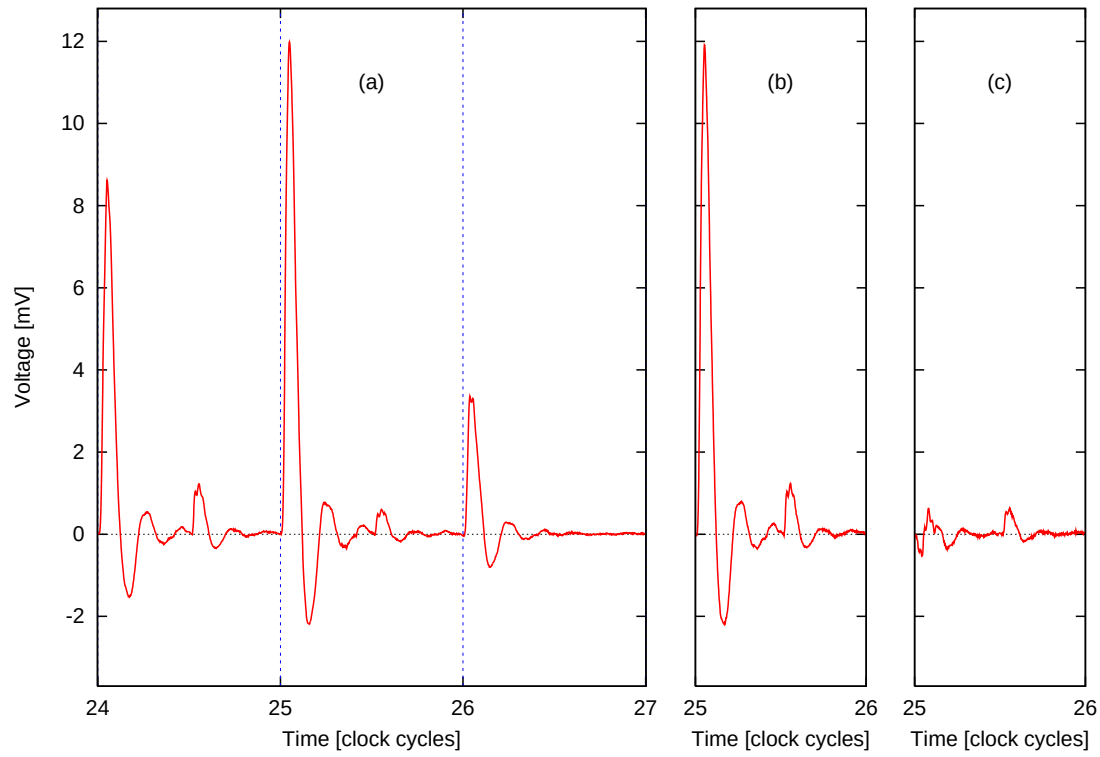


Figure 3.32: (a) Differential traces for $i = 8$, (b) Sum $\Delta_{24} + \Delta_{26}$ of the periods 24 and 26, (c) Difference $\Delta_{25} - (\Delta_{24} + \Delta_{26})$.

Note: The **left** half of $LR_i \oplus LR_{i+1}$ is equal to **$R_{i-1} \oplus R_i$** because $L_i = R_{i-1}$ and $L_{i+1} = R_i$. Identically, the **right** half of $LR_i \oplus LR_{i+1}$ is equal to **$L_{i+1} \oplus L_{i+2}$** because $R_i = L_{i+1}$ and $R_{i+1} = L_{i+2}$.

Given that DES leaves unchanged one half of the datapath after every round, there exists a correlation between the previous and the next round when using the HD weighting function. The transfer during the clock cycle 24 is $R_{i-1} \rightarrow R_i$, taking place “physically” in the register R, whereas in the next clock cycle 25, it occurs in the register L. The glitching activity of XORK (Eqn. 3.7) and XORL (Eqn. 3.8) respectively add up to those register transfers. The same remark applies for the $L_{i+1} \rightarrow L_{i+2}$ transfer. But eventually, the 64-bit transfer $LR_i \rightarrow LR_{i+1}$, realized at clock 25 in the LR register, is equal to the sum of the transfers in R and L at clocks 24 and 26, in a virtual RL register. As the plaintext is random, the data in registers L_i and R_i have the same statistical properties, which explains why an activity in the real LR register cannot be distinguished from that of an artificial RL register.

The Fig. 3.32(a) also shows a peak exactly in the middle of the clock periods 24 and 25. It represents the sampling of the new data in the DFFs head latch, that toggles states synchronously with the falling edge of the clock. The amplitude of the peak is small, compared to that corresponding to register transfers, because the head latch is little capacitively loaded: it singly drives an internal net of the DFF gate. The curves were acquired while the DES module was running at the low frequency of 32 MHz, although it was synthesized for a target frequency greater than 100 MHz. The new value LR_{i+1} is thus already valid 15.625 ns after the rising edge of the clock. This justifies that the dissipation of the input latches of the LR DFFs is correlated to the weighting function (3.11).

The peaks caused by the negative edge of the clock follow a similar conservation property as the one observed regarding the LR value. On one graph $i \in [1, 13]$ of Fig. 3.31, the sum of those at clocks $i - 1$ and $i + 1$ is equal to that of the clock i .

3.6.2.2 Interpretation of HD at rounds [1,2]

The differential curves for $i = 0, 1$ have a “combinatorial” peak in the first clock period of the encryption. It has been made clear in Sec. 3.5 that a peak occurring not after a rising edge of the global clock is a coherent event triggered by an edge on one control signal that modifies the datapath. Typically, those signals are multiplexors selection inputs. In round 16, the two MUXLR selection signals `sel_swap` and `sel_ext_not_fbk` have a rising edge. Notice that the reason for the delay of the peak w.r.t. the rising edge of the clock is that the selection signals are computed externally by a control block, that is in turn synchronized with the clock. The delay is thus a propagation time. Besides, the selection signal may have glitches, which may explain the fact that the peak is not sharp: it is wider than that caused by the (glitch-free) clock.

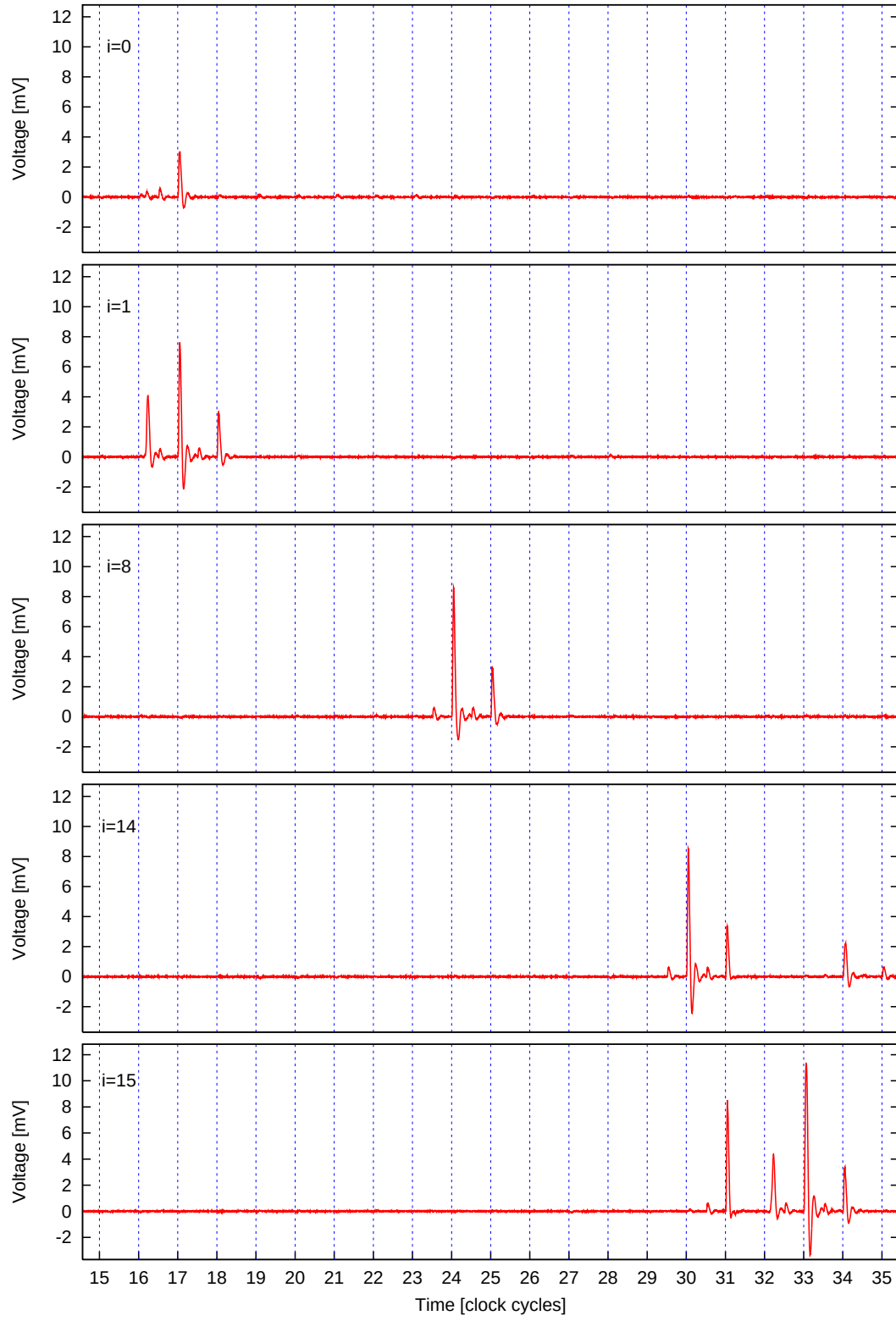
The origin of the dissipation correlated to (3.11) induced by the selection signal is explicited by the splitting into the **left** and the **right** halves:

$$|L_i \oplus L_{i+1}|, \quad (\text{in Fig. 3.33}) \quad (3.12)$$

$$|R_i \oplus R_{i+1}|. \quad (\text{in Fig. 3.34}) \quad (3.13)$$

Now, as $R_i \oplus R_{i+1} = L_{i+1} \oplus L_{i+2}$, the two differential traces become identical when $i \leftrightarrow i + 1$. This is a general property of Feistel networks: the differential trace for the whole LR datapath is equal to the superimposition of that of R and that of L delayed by one round.

Figure 3.35 introduces three intermediate nodes `tmp{1,2,3}` involved in the signature of the transition $LR_0 \rightarrow LR_1$. An analysis on clock period 16 is given in Tab. 3.15. In this table, “swap” and “fbk/ ext” are short for “sel_swap” and “sel_fbk_not_ext”.

Figure 3.33: Differential trace for the selection function $|L_i \oplus L_{i+1}|$.

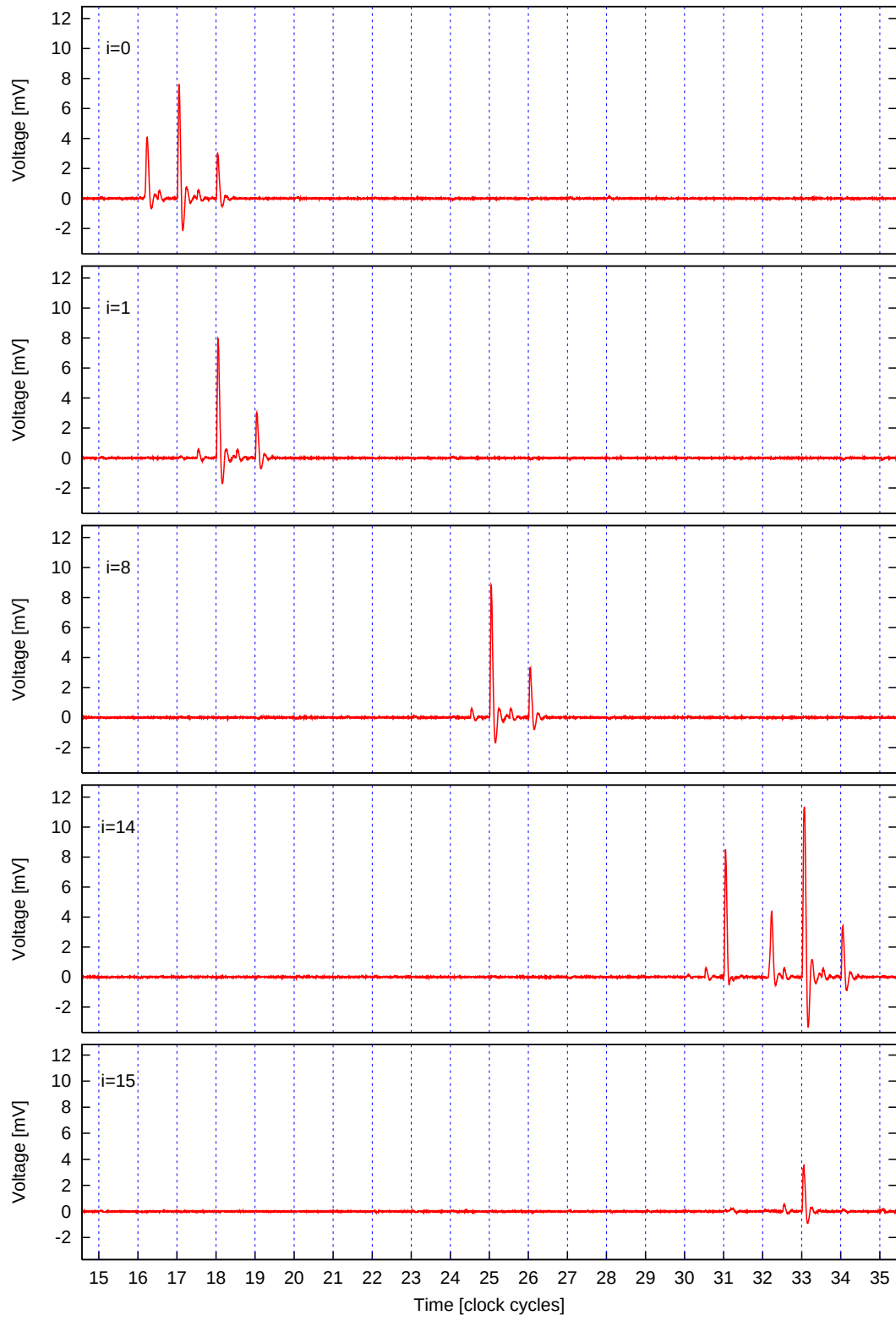


Figure 3.34: Differential trace for the selection function $|R_i \oplus R_{i+1}|$.

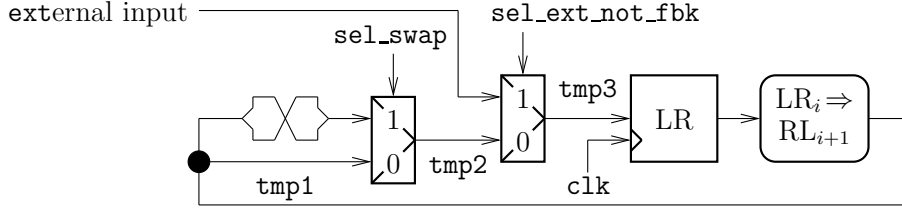
Figure 3.35: Simplified datapath involved in the $LR_0 \rightarrow LR_1$ selection function.

Table 3.15: Events in the datapath of Fig. 3.35 between clock cycles 16 and 17.

Event	initial	clk $\uparrow$ (16)	$R_0 \rightarrow R_1$	swap $\uparrow$	fbk / $\overline{\text{ext}}\uparrow$	clk $\uparrow$ (17)
LR ^a	XX	LR_0	LR_0	LR_0	LR_0	LR_1
tmp1	XX	XL_1 ^b	RL_1	RL_1	RL_1	R_1L_2
tmp2	XX	XL_1	RL_1	LR_1	LR_1	L_1R_2
tmp3	LR_0	LR_0	LR_0	LR_0	LR_1	LR_1
Transition	$\emptyset$	$\emptyset$	$\emptyset$	$2 \times (3.13)$	$(3.12 \ \& \ 3.13)$	$(3.12 \ \& \ 3.13)$

^aNotice that the register LR is updated synchronously with every rising edge of clk.^bThe evaluation of $L_1 \doteq R_0$ happens instantaneously after the rising edge of the clock.

3.6.2.3 Interpretation of HD at rounds [15,16]

In Sec. 3.5.3, it has been explained that in the case of the cryptographic algorithm DES, the key can be traced by power analysis throughout the algorithm (Sec. 3.5.3.1), because of the simplicity of the key schedule, but that the message is instantaneously decorrelated (Sec. 3.5.3.2.) The analysis at rounds [3,14] shows that, because of the Feistel structure, the correlations are actually kept one round before and after the selection function. Now, the differential traces at round [15,16] disclose correlations over five clock periods. Such a high longevity is fairly unexpected.

This observation deserves an in-depth explanation. The profusion of curious correlation peaks accentuates the information leakage, thus fostering sneak attacks.

The contents of the LR register after the encryption, is shown to remain consistent until two cycles past the computation end. The Tab. 3.16 presents this results. Based on these facts, a register transfer analysis is given below:

- **at clock #32**, the transition signature is $|LR_{15} \oplus RL_{15}|$, which results in no peak in neither Fig. 3.33 nor Fig. 3.34.
- **at clock #33**, the transition $R_{16}L_{16} \rightarrow L_{16}L_{15}$, yields both a signature in register L at $i = 15$ ($R_{15} \oplus R_{16} = R_{16} \oplus L_{16}$) and in register R at $i = 14$ ($R_{14} \oplus R_{15} = L_{16} \oplus L_{15}$.)
- **at clock #34**, the transfer $L_{16} \rightarrow L_{15}$ signs for the transition $|R_{14} \oplus R_{15}|$ in register L at $i = 14$.

The peak in Fig. 3.34 for $i = 14$ at clock #33 is larger than other similar peaks for $i < 14$. The context for this peak is different from the other $R_i \rightarrow R_{i+1}$ transfers occurring during the encryption.

- In clock cycles [16,32], the round key is updated at every round. Thus, when R_i is updated into R_{i+1} , the difference $R_i \rightarrow R_{i+1}$ propagates through the R register, the XORK gates,

Table 3.16: Register LR contents after one SecMat V1 DES-ECB encryption.

Clock	Round	LR contents	Combinatorial round logic output	Swap $L \leftrightarrow R$?
$\vdots$	$\vdots$	$\vdots$	$\vdots$	yes
30	14	LR_{14}	RL_{15}	yes
31	15	LR_{15}	RL_{16}	no ^a
32	16	RL_{16}	$(L_{16}, R_{16} \oplus f(L_{16}, K_{16})^b)^c = (L_{16}, L_{15})$	yes
33	17	(L_{16}, L_{15})	$(L_{15}, L_{16} \oplus f(L_{15}, K_{16})^d) = (L_{15}, XX)$	yes
34	18	(L_{15}, XX)	$(L_{15} \oplus f(XX, K_{16}), XX) = (XX, XX)$	yes
35	19	(XX, XX)	(XX, XX)	yes
$\vdots$	$\vdots$	$\vdots$	$\vdots$	yes

^aThe last round is the sole occasion where the datapath two halves are not swapped.

^bThe key is unchanged after the encryption (*i.e.* `sel_not_shift=1`.)

^c $f(L_{16}, K_{16}) = f(R_{15}, K_{16}) = R_{16} \oplus L_{15}$: the comeback of this value is highly astonishing.

^dThe value of $f(L_{15}, K_{16})$ is decorrelated from any L_i and R_i ; it can be considered as a random mask “XX”.

and substitution boxes. However, the internal nets of the sboxes have been decorrelated by the sub-key update.

- After clock cycle #32, the round-keys are always the same: $PC2(CD_{16}) = PC2(CD_0)$. Thus when the value R_{15} is replaced by the value R_{14} , the internal nets of the sboxes have kept a strong dependency in R_{15} . The correlation continues thus deeper in the logic. A quantitative evaluation of the correlation gain requires an accurate knowledge of the netlist. A static simulator could be an relevant tool to provide more insights on the “entanglement” of the sbox with its input. This assertion is validated by the computation of the difference between a “regular” and the “clock cycle #33” peaks for the $R_i \rightarrow R_{i+1}$ transfer. The resulting waves for two different acquisition campaigns are shown in Fig. 3.37. The waves shapes differ because the two campaigns have been led on two different acquisition boards. The graph shows that the difference begins to sign about 1.20 nanoseconds after the rising edge of the clock, which proves that extra dissipation is caused by the activation of combinatorial gates.

The effect just mentionned should not manifest if all the round keys happen to be equal. The plot shown in Fig. 3.36 compares the maximum peak height for differential waves obtained from the selection function (3.12) for two acquisition campaigns:

1. one with a weak key ($\{0x00\}^8$) and
2. another with a variable (random) key.

During the encryption (15 register transfers), peaks obtained with the null key are the highest. The reason for the increase trend from $i = 1$ to 15 of the peaks amplitude is still unknown². However, after the encryption, at clock period 33, the peak corresponding to the transfer $L_{16} \rightarrow L_{15}$ is the same, as expected.

The former analysis showed that keys that do not vary randomly at every round enhance the power attacks. In the case of DES, those keys are for instance weak and semi-keys, as well as 48 potentially weak keys [22].

The last peak whose presences remain unexplained is:

1. The combinatorial peak in Fig. 3.33 for $i = 14$ at clock #34. This peak is the only one to disappear if the key is randomized.

²It just suggests that an known ciphertext DPA is likely to be more powerful than a chosen plaintext DPA.

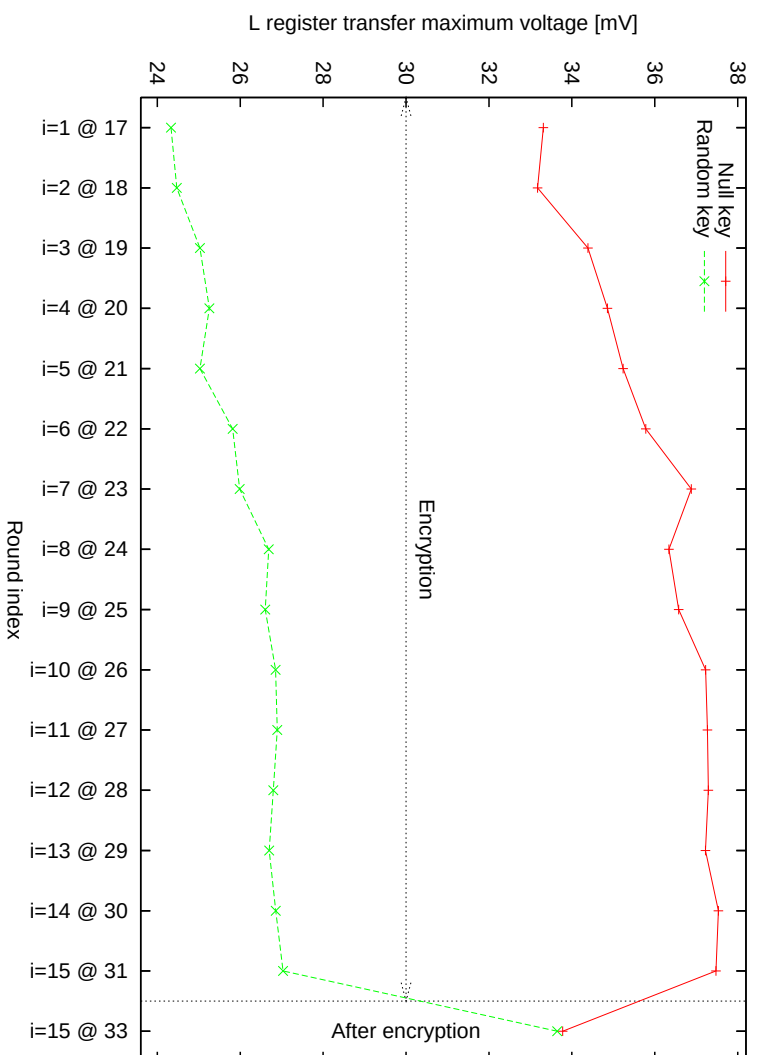


Figure 3.36: Amplitude of differential waves computed with the selection function (3.12).

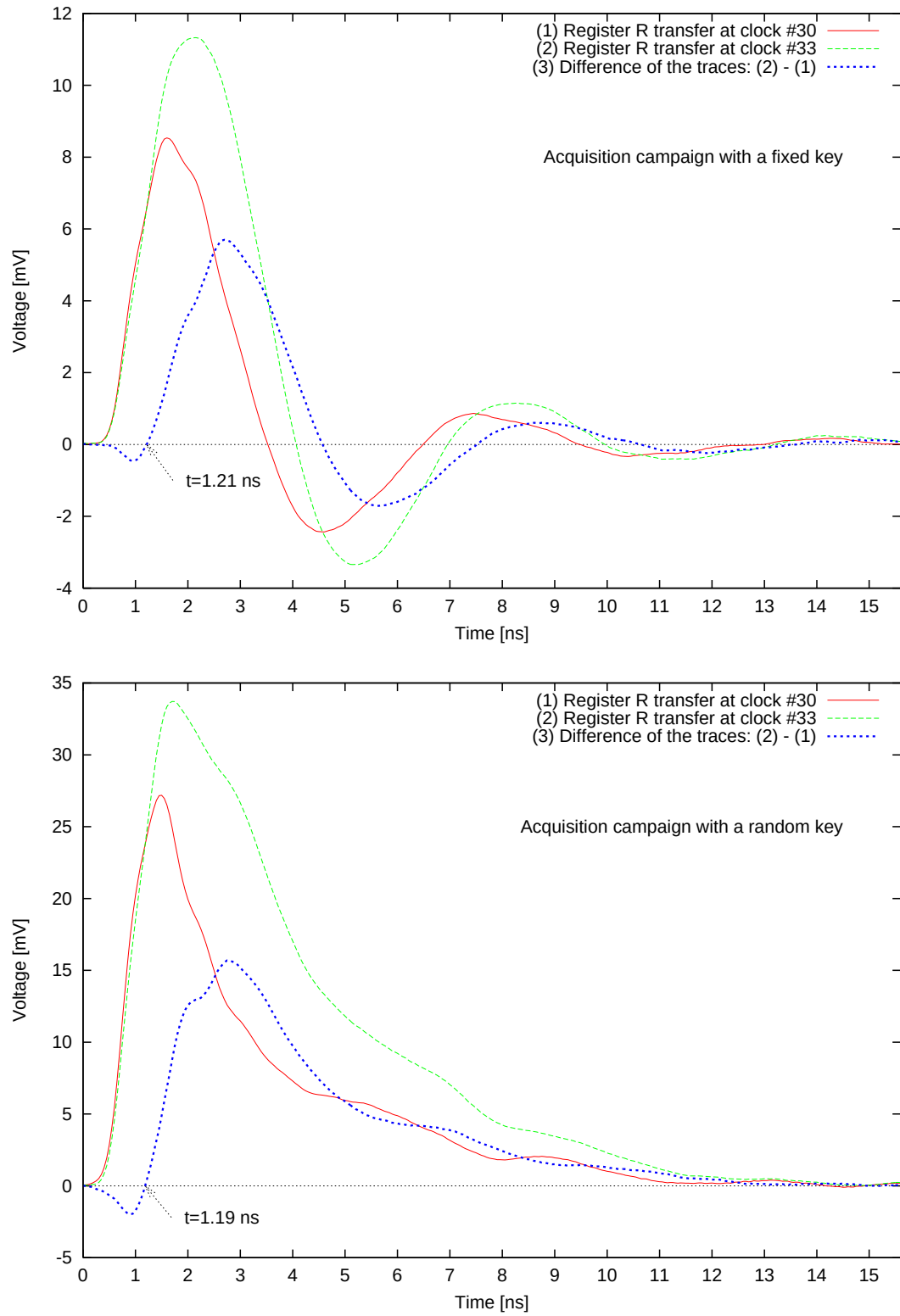


Figure 3.37: Difference of differential traces for a register transfer in R, at clock periods #30 (during the encryption) and #33 (after the encryption.) Only the first half period is plotted (the full clock period lasts $\frac{1}{32 \text{ MHz}} = 31.25 \text{ ns.}$)

Table 3.17: Analysis of the glitches dissymmetry in XORL at the end of clock cycles 31–33.

Clock #31	Clock #32	Clock #33

The chronology provided in Tab. 3.16 shows the survival of correlated round messages L_i and R_j , where $i, j \in \{15, 16\}$. This explains the differential trace obtained using the Hamming weight selection function (3.6) with $i = 15$. As expected, in clock period [31,32], the sbx outputs are betrayed (albeit with a poor signal quality) by the power analysis. At the beginning of clock period 32, Neither of transitions $|L_{15} \oplus L_{16}|$ (Fig. 3.33) nor $|R_{15} \oplus R_{16}|$ (Fig. 3.34) sign. On contrary, as explained in Tab. 3.16, the transition $LR_{15} \rightarrow RL_{16}$ signs a Hamming weight equal to $|L_{15} \oplus R_{16}|$, occurring in register L. This leads to the sequential peak observed in Fig. 3.30 for $i = 15$ at clock 32. Now, in these differential traces, two peaks remain to be explained:

1. a small peak following the sequential peak just identified at clock 32 and
2. another small peak at clock 32.

These two peaks are due to a glitch propagation through XORL, as already detailed in Sec. 3.6.1. The state of the datapath at the end of clock cycles 31, 32 and 33, when every signal has stabilized to its definitive value, is given in Tab. 3.17. In this table, the input b is slow, since it is on the critical path: effectively, both the round key and the sbx must be computed to present a valid input to port b of XORL. The input a is fast, because it is directly the output of the datapath register L. This table shows that the steady input b take the following values:

- At the rising edge of clock 32, b remains transiently equal to $R_{16} \oplus L_{15}$.
- At the rising edge of clock 33, b remains transiently equal to $L_{15} \oplus R_{16}$.
- At the rising edge of clock 34, b remains transiently equal to $f(R_{15}, k_{16}) = XX$.

As a consequence, two glitches are produced for the $|f(R_{15}, K_{15})| = |R_{16} \oplus L_{15}| = |L_{15} \oplus R_{16}|$ selection function, at clock 32 and 33.

3.6.3 Single *versus* multi-bit HW or HD selection functions

The analyzes of the differential traces obtained by the weighting with an HW (*resp.* HD) selection function on the full 64-bit register LR has been conducted in Sec. 3.6.1 (*resp.* Sec. 3.6.2.) The origin of the correlation peaks have been shown to be of three types:

1. **state** at the output of a combinatorial logic cone (peaks in clock period $16+i$ in Fig. 3.30),
2. coherent **glitching activity** of an XOR gate (peaks in clock period $16+i$ in Fig. 3.30),
3. **transition** of the output of a sequential gate, synchronized either with the clock or with a control signal (all the peaks in Fig. 3.31.)

The goal of this section is to show that, under normal conditions, the **state** (item no. 1 of the previous list) of a net is an invalid selection function. The evidence is provided experimentally,

Table 3.18: Selection functions used for differential waves plots in Fig. 3.38.

	State selection (HW)	Transition selection (HD)
Single-bit	$ S_{HW} \& (0x80, 0x00^3) = S_{HW} _1$	$ S_{HD} \& (0x80, 0x00^3) = S_{HD} _1$
Multi-bit	$ S_{HW} $	$ S_{HD} $

thanks to four differential traces. The four differential traces use either one or all the 32 bits from the two functions:

$$\begin{aligned}
 S_{HW} &\doteq f(R_0, K_1), & (\text{state of the sboxes output at round } \#i = 0) \\
 S_{HD} &\doteq f(R_0, K_1) \oplus f(R_1, K_2). & (\text{transition of the sboxes output at round } \#i = 1)
 \end{aligned}$$

The actual selection functions, and the resulting differential traces, are given in Tab. 3.18. The multi-bit Hamming weight function has already been plotted in Fig. 3.30($i = 0$) and studied in previous Sec. 3.6.1.

Not surprisingly, the single and multi-bit traces exhibit peaks at roughly same dates. Nevertheless, the peaks are not all **homothetic**; a factor 32 is indeed expected. The fact that a peak scales (or not) when adding up bits of a same multi-bit selection function reveals whether the function matches an elementary side-channel. The underlying criterion is the notion of **physical significance** of a selection function.

Definition 6 (Physical significance of a selection function) *A selection function is physically significant when its logical expression accurately models a physical dissipation.*

When a single-bit selection function is physically significant, it typically matches the dissipation of one target net of a circuit. Given that the energy is an extensive quantity, the power dissipation of a collection of nets simply adds up; this is true at least at first order, *i.e.* if the nets are not physically cross-coupled (generally via undesirable parasitic capacitances.) Now, let us consider a collection of nets that are not logically cross-coupled: this situation is representative of a cryptographic datapath, where the bits are made as independent from each other as possible, from a statistical point of view. If the multi-bit selection function for such a collection of nets is physically significant, then the power peak is equal to the coherent sum of the individual peaks for every net within the collection.

Based on this rationale, the peaks in Fig. 3.38 are analyzed. In the first round, only the HW selection functions have peaks. They represent the item no. 1: *state-wise* dissipation. In the second round, peaks are caused by items no. 2 & 3. They can be grouped under the common denomination of *activity*-related dissipation (either “differential activity” for item no. 2 or “simple activity” for item no. 3.) The maximum amplitude of the peaks are reported in Tab. 3.19. It can be seen that the peaks of the first round do not scale, whereas others do. These two constataion are expressed in properties 5 and 6.

Property 5 (Physical insignificance of HW) *The Hamming weight (HW) selection functions are physically insignificant w.r.t to the targeted net(s.) These functions can be significative of an asymmetry of the HD activity at the output of gate (such as the **glitching activity** of an XOR gate.)*

Property 6 (Physical significance of HD) *The Hamming distance (HD) selection functions are physically significant: they model the dissipation of the targeted net(s.) Without caring about physical units, this dissipation is indeed assessed by the amount of changes (*i.e.* **transitions** or activity) of the nets value.*

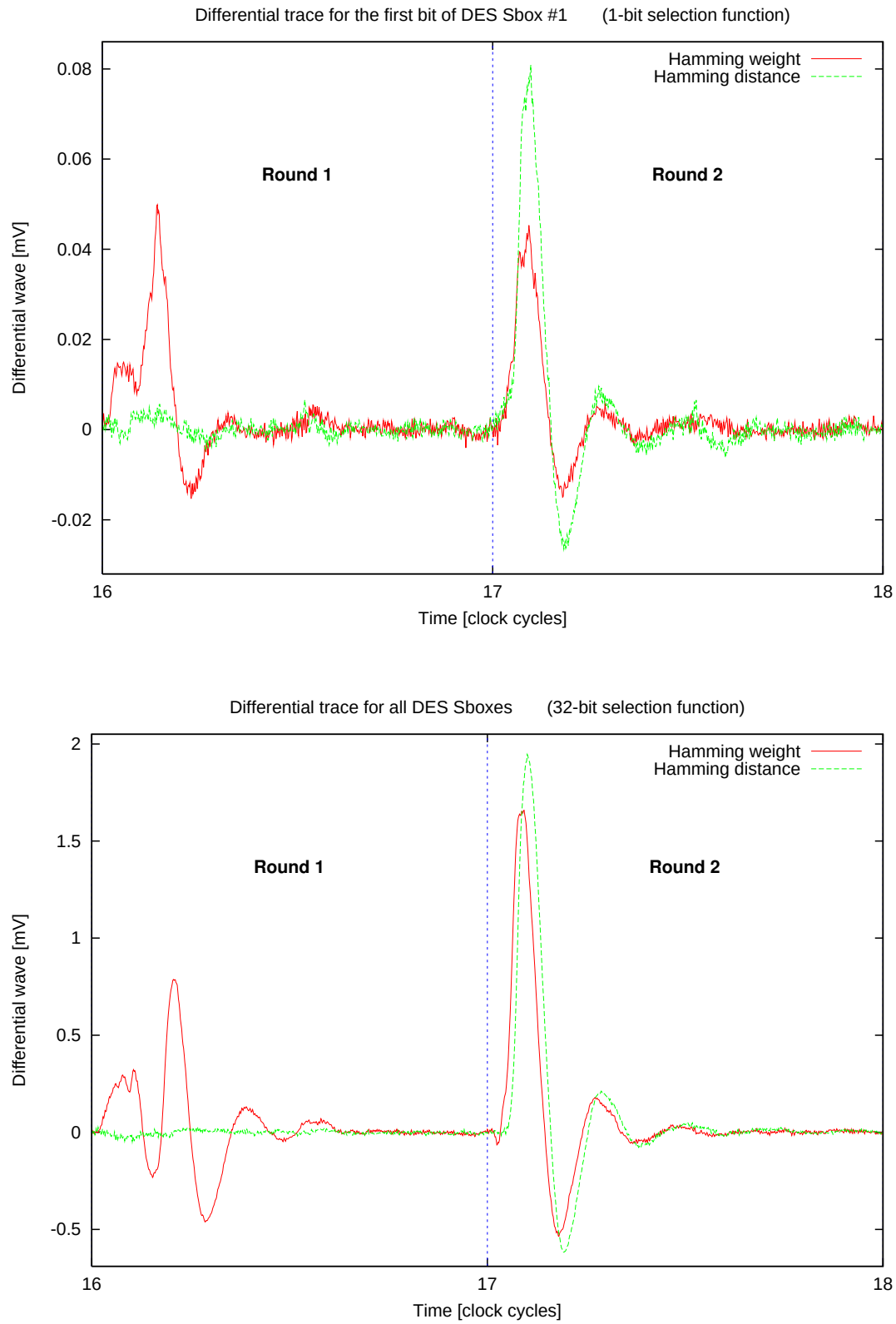


Figure 3.38: Single-bit (*upper*) and multi-bit (*lower*) selection functions comparison for DES sboxes outputs Hamming weight or distance, as per Tab. 3.18.

Table 3.19: Maximum amplitude of the peaks of differential traces shown in Fig. 3.38.

Round 1			Round 2		
	HW	HD		HW	HD
Single-bit	0.050 mV	0.007 mV [NA]	Single-bit	0.045 mV	0.081 mV
Multi-bit	0.787 mV	0.029 mV [NA]	Multi-bit	1.659 mV	1.945 mV

The fact that the Hamming weight signs all the same is probably due to a **degenerescence** of the selection function. If an initial or final state is constant, then the HD is brought back to an HW. The complex logic that makes up a substitution box can, by chance, make such a situation appear with a low probability. In such cases, the HW selection leads to a “near-match”, hence causing a peak. This explanation is very relevant for software implementations, where it is not unlikely that a register contain a deterministic value prior to being loaded with the targeted word (thus turning HD into HW.)

It can be noticed that the best way to estimate the output change of a combinatorial function, such as sboxes, remains the HD. The corresponding peak in clock period 1 is slightly delayed w.r.t. the XORL peak. However, the delay is very small, and certainly accounts only for the first glitches at the output of the sboxes. For every sbox output, the first glitch wipes out the previous value. Following glitches (more scattered in time) match the transitions towards the final value of the sboxes. This explains why the HW differential traces peaks occur latter in the clock period than HD’s. Those glitches are indeed revealed by an HW-type selection function. Unfortunately, only the last glitch stabilizes to the HW. This glitch temporal occurrence does depend both on the data and of the targeted bit. The large dispersion of the last glitch accounts for the inconsistency of the HW selection function.

Property 6 shows that, from a theoretical point of view, HW is not suitable to model attacks. Attacks using HW that happen to be successful succeed in fact thanks to:

- either the exploitation of an happy degenerescence,
- or of a secondary HD effect, such as the asymmetrical glitching activity of an XOR gate.

3.6.4 Conclusion: improvement of side-channels analyzes

The traces analyzes accounted for in this Sec. 3.6 are very specific to the DES co-processor architecture embedded in SecMat V1. However, their interpretation has led to the identification of new vulnerabilities.

First of all, the multiplicity of the peaks (see for instance the triples in Fig. 3.31, for $i = 0$) can be exploited to increase the leakage signal. In particular, on Feistel-networks, additional peaks can appear after the encryption is over (until clock period #34), unless the register LR is frozen. The corollary is that a conventional DPA on the last round does succeed. This opens a serious security breach, that is made possible by an optimization: if the LR register had an enable, then:

- the critical path would be slightly longer, thus impeding the co-processor speed, but
- the dreaded register transfer $LR_{15} \rightarrow LR_{16}$ would never manifest (refer to Sec. 3.7.)

We also noted that the falling edge of the clock contributes to the power signature.

Then, we observed that glitches can be exploited too. We proved that the insulation of a glitch traversing an XOR gate is done thanks to the Hamming weight of the steady input.

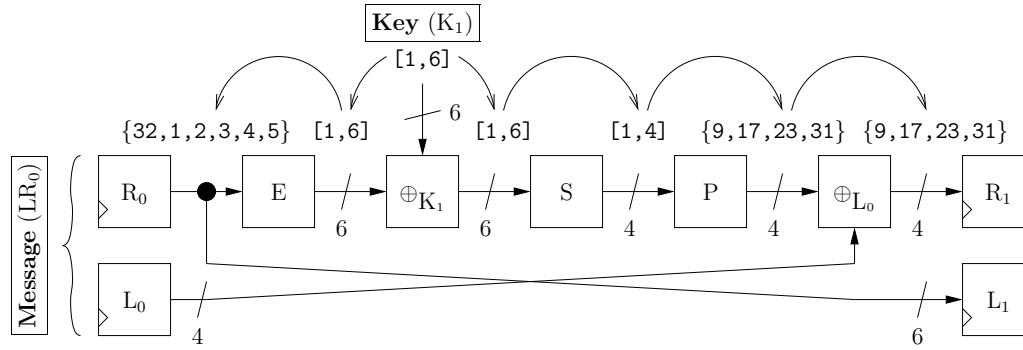


Figure 3.39: Datapath of DES involved in the DPA attack of the first round.

These observations demonstrate the possibility to realize “*multi-selection function*” side-channels analyzes. We are currently working to quantify the improvement conveyed by this new analysis tool.

3.7 Realization of the DPA on DES

The DPA is performed on the first round, because the last round is a transfer from register LR to register IF. The IF register contains meaningless data, hence the unexploitability of the transfer. Consequently, the attack is a known-plaintext DPA on the iterative hardware DES cryptoprocessor described in Sec. 3.2.6 at page 65. The side-channel that is considered is the register transfer $LR_0 \rightarrow LR_1$ between rounds 1 and 2.

Notice that in the DES module of SecMat V1, the DPA surprisingly **does work** on the last round. The reason is a consequence of the results obtained in Sec. 3.6.2. As shown in Tab. 3.16, the register L contains the values:

- R_{16} at clock period #32 and
- $L_{16} = R_{15}$ at the following clock period.

The fateful transfer $R_{15} \rightarrow R_{16}$ thus happens, albeit:

- not in this order, but in the other,
- not at clock period #32, but at clock period #33,
- not in register R, but in register L.

3.7.1 Selection function for the DPA on the DES architecture of SecMat V1

Given that DES is a Feistel network, $L_1 = R_0$. As this transition does not depend on the key K_1 , it does not leak any sensitive information. However, The transition $R_0 \rightarrow R_1$ is relevant. The datapath for this transition is depicted in Fig 3.39.

Property 7 (DES diffusion) *None of the DES sboxes clobber the registers containing the data previously holding their inputs. Put differently, the set of indices of the plaintext that, mixed with a subkey, yields the sbox input is disjoint from the sbox output.*

Table 3.20: Input / output along one DES round for the eight sboxes (For the sake of clarity, the functions $\oplus_{K_1}$ and $\oplus_{L_0}$ have been omitted because they do not reorder the wires.)

Sbox	R_0	E	S	P
# 1	{32, 1, 2, 3, 4, 5}	{1, 2, 3, 4, 5, 6}	{1, 2, 3, 4}	{9, 17, 23, 31}
# 2	{4, 5, 6, 7, 8, 9}	{7, 8, 8, 10, 11, 12}	{5, 6, 7, 8}	{13, 28, 2, 18}
# 3	{8, 9, 10, 11, 12, 13}	{13, 14, 15, 16, 17, 18}	{9, 10, 11, 12}	{24, 16, 30, 6}
# 4	{12, 13, 14, 15, 16, 17}	{19, 20, 21, 22, 23, 24}	{13, 14, 15, 16}	{26, 20, 10, 1}
# 5	{16, 17, 18, 19, 20, 21}	{25, 26, 27, 28, 29, 30}	{17, 18, 19, 20}	{8, 14, 25, 3}
# 6	{20, 21, 22, 23, 24, 25}	{31, 32, 33, 34, 35, 36}	{21, 22, 23, 24}	{4, 29, 11, 19}
# 7	{24, 25, 26, 27, 28, 29}	{37, 38, 39, 40, 41, 42}	{25, 26, 27, 28}	{32, 12, 22, 7}
# 8	{28, 29, 30, 31, 32, 1}	{43, 44, 45, 46, 47, 48}	{29, 30, 31, 32}	{5, 27, 15, 21}

This characteristic holds by construction for DES: the outputs of every sbox are injected into other sboxes, but the current one. The property 7 can be proven exhaustively, as shown in Tab. 3.20: the intersection between the first and the last column is always empty. The cryptological concept expressed by the property 7 is that of “diffusion”.

The transition that is exploited (for sbox #1) is:

$$R_0\{9, 17, 23, 31\} \rightarrow L_0\{9, 17, 23, 31\} \oplus S_1(R_0\{32, 1, 2, 3, 4, 5\} \oplus K_1[1, 6]) . \quad (3.14)$$

Using as weighting function the XOR between the initial and the final values in Eqn. (3.14), the DPA is performed on a set of 50 000 traces. The attack is successful starting from about one thousand traces. Much fewer traces are required if the maximum peak criteria exposed in Sec. 2.3.5.1 is traded for a maximum likelihood estimation [86].

The signal-to-noise ratio, as defined in Eqn. (2.7) of the DPA for all the substitution boxes is given in Appendix B.2 at page 163. The theoretical value of the SNR is reported on the graph. It is evaluated according to the model Eqn. (2.9), with the two following modifications:

1. the signal (for the correct key guess) is not included in the denominator, because it does not contribute to the noise, and
2. it is adapted to DES, as per Eqn. (3.14.)

The asymptotical measured SNR match the predicted value, as shown in Fig. 3.40. In this histogram, the SNRs are given without units.

3.7.2 DPA on traces integrals

Definition 7 (Trace integral) *A trace integral is a scalar that represents a selected portion of a full trace. Typically, the integration of a trace in a window yields a trace reduced to one point, that is a possible candidate for a trace integral.*

The DPA is performed on three types of traces integrals:

1. integral over the acquisition period (including parts of data loading and unloading),
2. integral over the sixteen rounds of the encryption and
3. integral over the first round of the encryption.

The characteristics of the three averagings are given below:

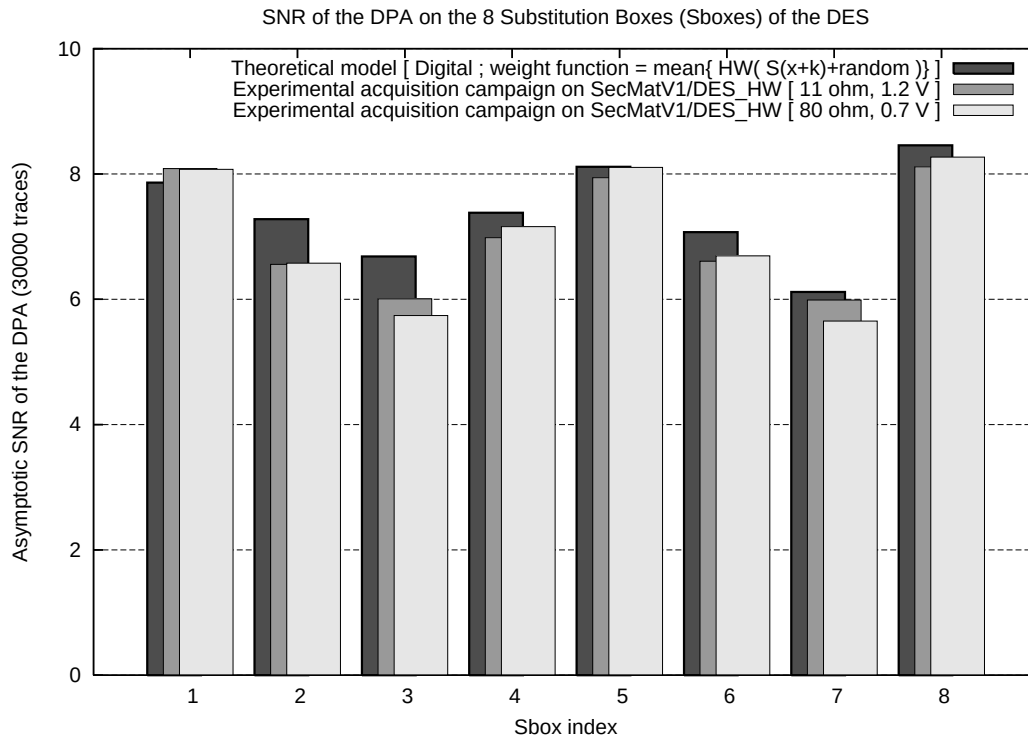


Figure 3.40: Comparison between theoretical *digital* model presented in Sec. 3.7 and experimental *analog* measurements of the DPA Signal-to-Noise Ratio (SNR) on the secret key encryption algorithm DES [71] embarked in the SecMat V1 ASIC (cf. the layout of Fig. A.1(a)).

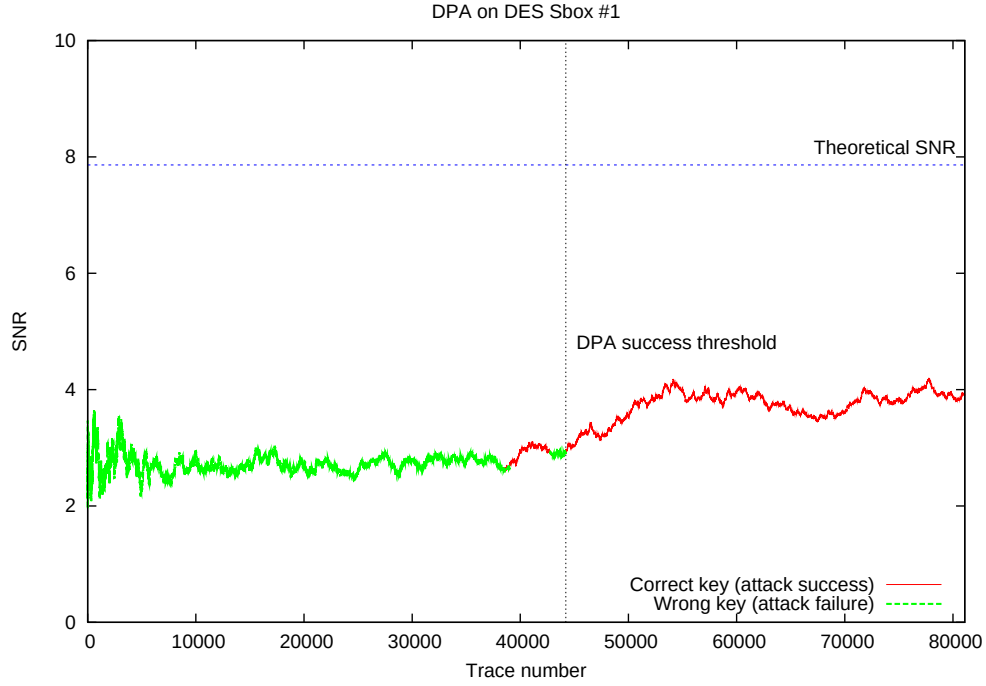


Figure 3.41: Hamming distance for sbox #1 on the scalar sum of the full power traces over 32 clock periods taking in one DES encryption.

1. The full traces have 20 000 data points, acquired at 20 Gsample/s. As the circuit runs at 32 Mhz, the full traces span over 32 clock periods. One representative trace is shown in Fig. 2.20; the integral consists in computing $\sum_{t=0}^{20\,000-1} \text{trace}(t)$.
2. For the second type of integral, the averaging is limited to the sixteen clock periods (clocks 16 to 32) taken by the DES encryption proper: only half of the number of points making up the trace are used in the summation.
3. The third type of integral consists in clipping the first sixteenth of the DES encryption, corresponding to one single clock period (clock 16 to 17.) The averaging is thus realized on 625 data points.

The DPA keeps on working for sbox #1, even with the averaging over the full trace, as shown in the figures 3.41, 3.42 and 3.43.

Apart from sboxes #3 and #4, all the sub-keys were retrieved with 81 089 traces, even with the harshest integral. The minimum number of traces to crack each sub-key is reported in Tab. 3.21.

It is interesting to remark that, without averaging (see SNR graphs at Appendix B.2), it is generally harder to find the correct key than with an averaging over the critical clock period of the first round. The reason is that the energy dissipated by the DFFs is spread, due to the signals modulation at 150 MHz. The summation over the clock period helps reconstructing the spread information.

The latter remark is justified by the fact that the attack is still faster when averaging on the first half of the period.

The recommendations that can be made for a security improvement is to use deterministic gates, because random delay insertions (see Sec. 4.1.2), *etc.* do not resist the temporal inte-

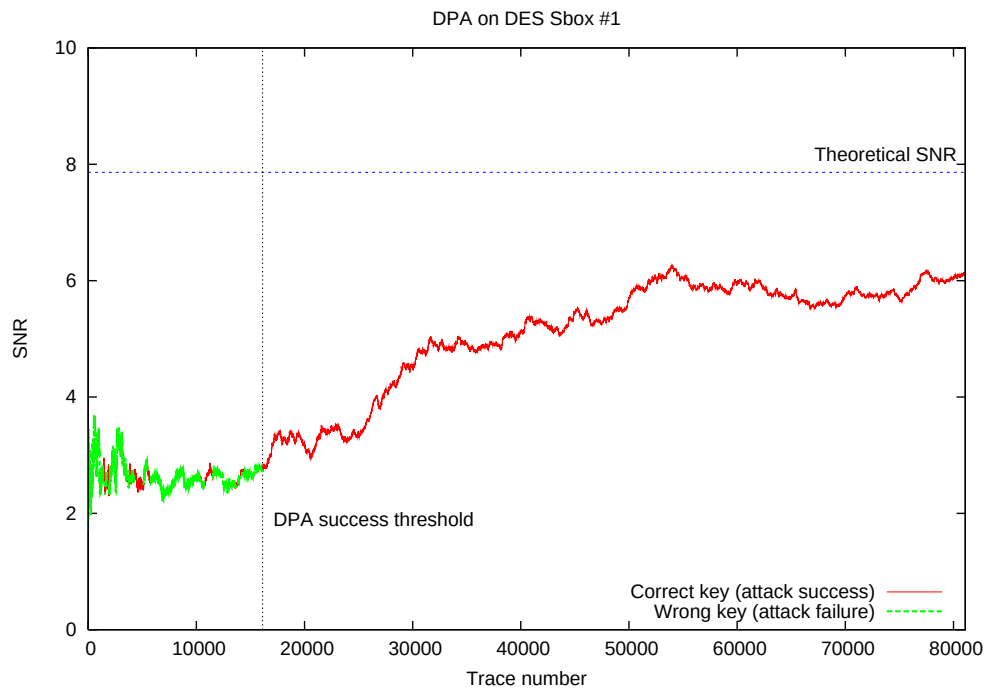


Figure 3.42: Hamming distance for sbox #1 on the sum of the traces points over the 16 clock periods making up a DES encryption.

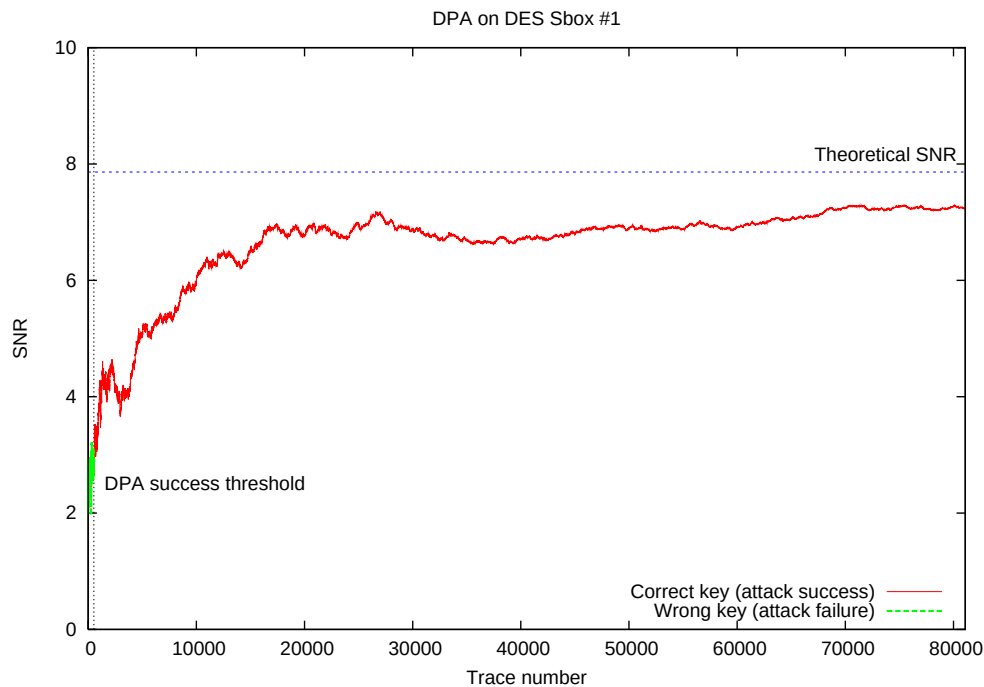


Figure 3.43: Hamming distance for sbox #1 on the sum of the data points inside of DES first round.

Table 3.21: Minimum number of traces to crack the unprotected DES iterative implementation.

Sbox	Average type			
	Whole trace	Whole DES	First round	No average
# 1	44 211	16 129	528	968
# 2	37 497	23 075	1 231	1 799
# 3	Failure	40 591	761	2 475
# 4	Failure	18 289	1 816	1 478
# 5	80 711	8 517	1 141	1 151
# 6	51 229	9 899	995	1 110
# 7	35 367	8 972	3 093	1 353
# 8	35 902	14 493	854	967

gration. Now, this section showed that even heavy integrations proved to be vulnerable to the DPA.

Chapter 4

Backend countermeasures against SCAs

The leakage identified in the previous chapter are inputs to devise recommendations for maximally protected implementations. The goal is to provide the designer with an insightful solution without any tradeoff, but that offers the best achievable security level. A standard cell library is presented, along with a place-and-route methodology to use it securely. The approach used in this proposed counter-measures is based on using indiscernability at the lowest possible level, namely the logical gate operating on individual bits of information.

4.1 Leaking no information

4.1.1 Information = distinguishability

First of all, it must be noticed that side-channel attacks are possible because the circuit's behavior can be observed without modifying its execution. This assumption is for instance incompatible with the quantum key distribution model. In such protocols, the security lays on the assurance that unauthorized observations can be detected. In BB84 [13], the potentially observation rate is estimated prior to agreeing on a secret. If the channel is private enough, the communication of key materials can begin *in clear*. Actually, cryptography exists because classical information can be copied verbatim without anybody being aware of the information theft. Charles Bennett goes even further [12]: the same arguments can be extended from *communication* to *computation*. As classical devices leak energy, a periodic refresh, implemented by the amplification properties of logical gates, is mandatory. In contrast, a quantum computer falls into either the *reversible* (see Sec. 2.2.1) or the *conservative* (see Sec. 2.2.2) computation styles. Bennett states that classic computation is merely quantum computation impeded by observability. And it can be added that classic computers are also more vulnerable, because of the existence of side-channels.

Now, we assume that any computers available nowadays leak information. Given two observable random variables D_0 and D_1 , we write $D_0 \sim D_1$ if D_0 and D_1 have the same distribution. Such a case is called “perfect indistinguishability”. A consequence is that for a given observation, there exists no evidence to decide whether D_0 or D_1 happened. For example, D_0 and D_1 cannot be distinguished by their mean, their standard deviation, or any higher moments. The use of indistinguishable phenomenon is thus a valid strategy, not to conceal the occurrence of an event, but to hide its nature.

This strategy can be enforced at high-level to fight timing attacks: one objective is to guarantee that the execution of a program requires an identical amount of time irrespectively

of the data being processed. A concrete contre-measure at C-language level is exposed in a work on the program counter security model [62]. At the assembly language level, Marc Joye proposes the side-channel atomicity model as a software balancing technique [14]. In the rest of this section, we are concerned with similar solutions, at the gate-level.

When modeling security at high level, few physical considerations enter into account. At the opposite, low level investigations lead to an abundance of possible side-channels. The balancing strategies are thus less straightforward; at the resolution of the clock period, two instructions can be balanced in time. Below the clock period granularity, the evaluation of every instruction does depend on the type of instruction; hence a variation of the power or the electromagnetic signature. As a consequence, low-level balancing strategies often require some approximations. For instance, the decorrelation of the activity of two nets will be possible, provided they are enough *spaced* or *shielded*.

To conclude, it must be mentioned that the EMA model is rampant. This attack might enable — but to what extent? — a localization of the measured dissipation. The ability to access local information defeats the counter-measures presented in this section.

4.1.2 Randomization

An alternative solution is to resort to the unpredictable randomization of the syndromes D_0 & D_1 to be distinguished. This idea is widely used in cryptography; the concept of initialization vector (IV), seed and nonce are very common, but aim to fight other threats.

In hardware design, digital randomization fall into two categories:

1. **data masking**, such as the duplication method [41], The technic involves a data randomization, like cleartext or key *blinding* [60]. that lowers the correlation between the data and the physical information leaked (its *syndrome*.)
2. **timing desynchronization**, as analyzed in the PhD of Fraïdy Bouesse [38]. The desynchronization often relies on a dedicated microelectronic design based on asynchronous logic with a dual rail datapath encoding [64, 109, 33].

Analog counter-measures also exist:

- One possible gate that inserts random delays is explicited in [94].

Anyway, the use of random number generators (RNGs) displaces the problem: the device to protect is now the RNG itself.

These counter-measure consists in adding noise; it is irrelevant in the scope of our analysis (*cf* Sec. 1.2.1), since it only increases the number of traces to acquire, but does not make the attacks impracticable. The successful realization of the DPA attack on traces averaged over one encryption (and even more) demonstrated in Sec. 3.7.2 indeed indicates that randomization strategies are not long-term counter-measures against SCAs.

When a run-time reconfigurable device is available, new *dynamical* strategies become possible. Some seminal ideas about **hardware obfuscation** and **adaptative resilience** are mentioned in [87].

In the rest of this chapter, our contribution for a *static* and *deterministic* transistor-level balancing counter-measure is presented. The principled counter-measure can be split into a computation cell (see Sec. 4.2) and guidelines to turn a regular design into a secure one (see Sec. 4.3.)

4.2 The secured library “SecLib”

SecLib has been designed to provide the highest possible level of immunity against SCAs, without compromise on the performances. The goal of the SecLib is not to be a definitive solution

against SCAs. The approach is rather a “proof-of-concept”. On the one hand, if the library proves to be secure, then more efficient libraries might exist with the same level of security, which is an advantage for the adoption of this technology in concrete designs. On the other hand, if the library fails to be secure, then it is very likely that no other comparable library will be. Other solutions against SCAs will thus need to be devised. Moreover, the high cost that we afford to pay in SecLib is justified by the fact that no viable solution against the SCAs has been put forward yet.

The effects of the characteristics variations across the wafer are not taken into account at that point. The rationale is that those variations can be reduced by over-sizing the design.

4.2.1 Secured standard cells

The motivation to specify a new cell library is that standard cells do not compute with a constant syndrome. The goal of this section is to come up with a balanced architecture, able to process all the inputs combinations in an indiscernability way.

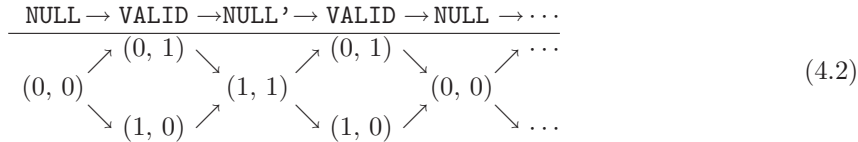
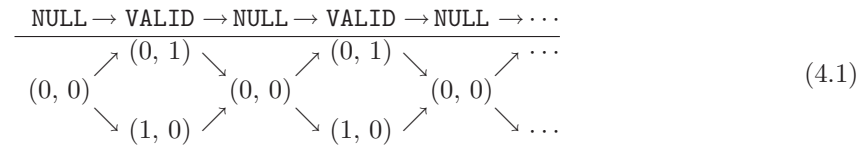
The difference between the rising and the falling transitions symptoms is a phenomenon making side-channel attacks possible. As a result, an electronic gate can only be claimed to be secured against side-channels attacks provided every logical change of this output correspond to the same type of transitions, *i.e.* always either a falling or a rising transition.

4.2.1.1 Dual-rail four-phase logic

One solution consists in encoding every data A on two wires, a_0 and a_1 , and to start every computation with the condition $a_0 = a_1$ satisfied. In practice, an event on either a_0 or a_1 represents the new valid data (0 or 1). The new state (further referred to as **VALID**) is characterized by $a_0 \neq a_1$. It is followed by a reinitialization step (further referred to as **NULL**), where the complementary wire is sent an event, so as to restart the new computation with a_0 and a_1 at the same electrical level. The **NULL** state can either be constant or alternative, as shown in (4.1) and in (4.2).

These schemes are referred to as *dual-rail four-phase logic* [33]. In those logics, the **NULL** must propagate: every gate whose inputs are **NULL** must produce a **NULL** output.

The constant **NULL** (say $\text{NULL} = (0, 0)$) scheme (4.1) is also known as *return-to-zero* (RTZ) signalization. As explained later on in Sec. 4.2.2, only RTZ is suitable with secured transistor-level structures. Additionally, the $(1, 1)$ state becomes forbidden and can therefore be used to detect faults [33, 64].



4.2.1.2 Secured cell micro-architecture

A gate with dual-rail inputs and outputs, using a succession of **VALID** and **NULL** states (4.1), starts any computation with rising edges, whatever the input data. Nevertheless, the gate must be further symmetrized to make its electrical behavior independent of its input data. The following three conditions are necessary for a constant syndrome gate:

1. The electrical network seen from input a_0 must be identical to the one seen from a_1 . If it was not the case, an event on a_0 would probably consume a different amount of power than an event on a_1 . A gate with multiple inputs, (a_0, a_1) , (b_0, b_1) , (c_0, c_1) , must remain electrically unchanged under any combinations of the transformations $a_0 \leftrightarrow a_1$, $b_0 \leftrightarrow b_1$, $c_0 \leftrightarrow c_1$.
2. The gate shall not produce anticipated outputs. If the gate evaluates before all the inputs are in the same state, either all **VALID** or all **NULL**, the computation duration depends on the input data. As this dependence is exploited by SPA, a secured gate shall wait for all its inputs before evaluating.
3. The gate must be memoryless. Every CMOS gate (but the **INV**) contains internal nodes that are not always driven. Undriven nodes store a state in their capacitance; the memorizing state is usually referred to as *high-impedance* or *Z* internal state. The memorization of an electrical charge from one computation to another leads to an observable conditional discharge. The discharge is a symptom, correlated to the gate input data sequence, and is for this reason exploited by side-channel attacks.

The first condition is straightforward on balanced truth tables gates, as for instance the inverter (**INV**), the exclusive-or (**XOR**), the half-adder (**HA**), the full-adder (**FA**) or the $2 \rightarrow 1$ multiplexer (**MUX21**). Although a universal logic set can be build out of those primitives (for instance $\{\text{INV}, \text{MUX21}\}$), it is worth trying to design a gate with unbalanced truth table, such as a **NAND**. As shown later in Fig. 4.1, dummy loads can be used to balance the electrical network. Additionally, as will become clear later, 3-input secured gates are much larger than 2-input gates. The 2-input **NAND** gate thus provides the most viable universal set.

The second condition can be implemented by a *rendez-vous* of the inputs, so as to make sure the gate processes the inputs (either **VALID** or **NULL**) only when they have all arrived. The inputs are thus first decoded by **C-Element** gates [99]. For instance, a 2-input (a_0, a_1) , (b_0, b_1) gate implements the rendez-vous between a_0 and b_0 , a_0 and b_1 , a_1 and b_0 , a_1 and b_1 . Only one of these rendez-vous produces an event. As soon as this event occurs, the gate can evaluate. Fig. 4.1 (inspired from [64]) shows a structure implementing a **NAND** that satisfies the first two conditions. The decomposition in transistors requires attention: the netlist must indeed preserve the conditions 1 and 2 (refer to Sec. 4.2.2.)

The proposed structure can clearly be divided into two parts:

1. the **front** part, in charge of the **synchronization** of the inputs,
2. the **back** part, a balanced combinatorial circuit in charge of the output **computation**.

This structure is by design *delay-insensitive*. In Fig. 4.1, the **OR** gates on the path to y_1 is dummy: its only purpose is to balance the y_0 and y_1 ways. More sophisticated architectures where the additional material is used for error detection (in the context of fault attacks) have been proposed [64].

The third condition can be fulfilled by adding circuitry to remove any conditionally undriven states. When a *Z* state occurs in the *P* transistor network, an appropriate *N* transistor logic can force its state to *VSS*. This method is illustrated on the three input **NOR** gate (**3NOR**) in Fig. 4.2. Another example of removal of *Z* states from the internal nodes truth table is shown for the **C-Element** in Fig. 4.3.

4.2.2 CMOS structures for the secured cells

The two parts, namely synchronization and computation, of the schematic of the **NAND** gate of Fig. 4.1 can be implemented independently.

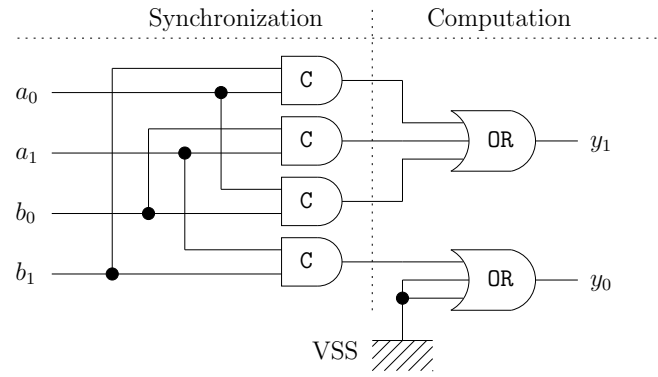
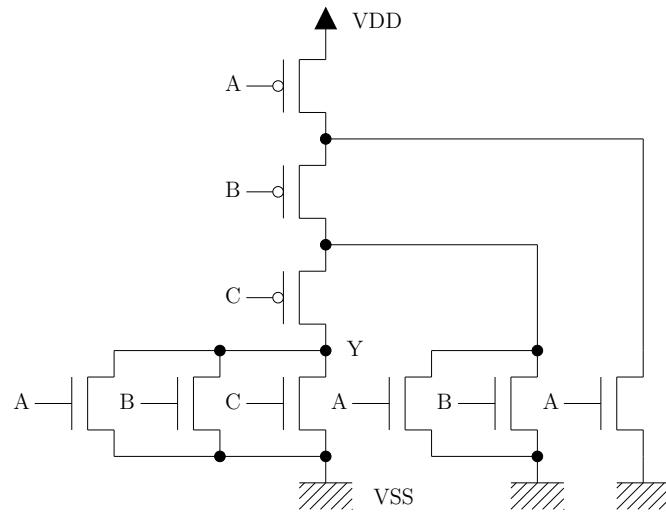


Figure 4.1: A secured NAND gate schematic.

Figure 4.2: Unbalanced – hence insecure – 3NOR gate, computing $Y = \overline{A + B + C}$, without internal Z states.

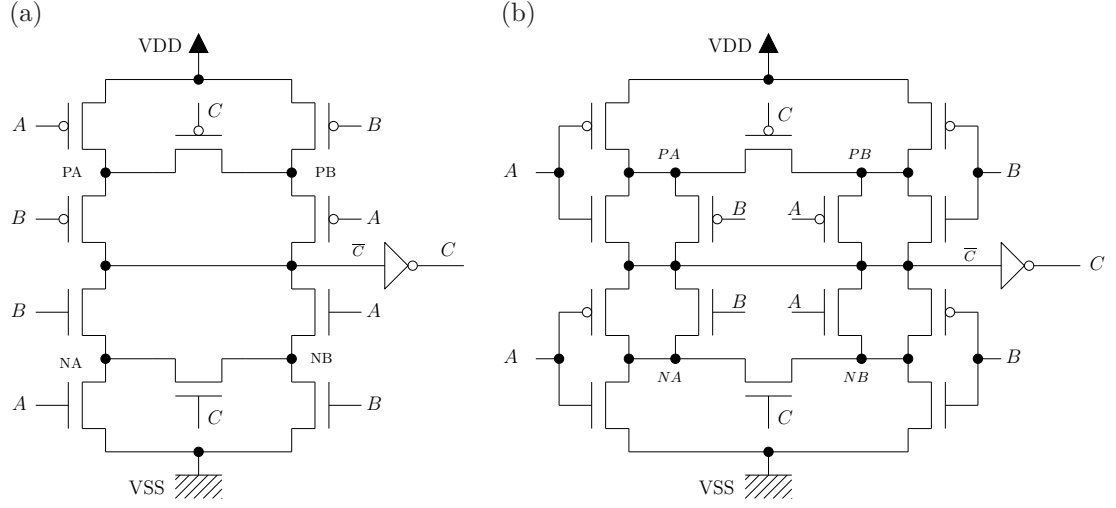


Figure 4.3: (a) *Symmetrical C-Element*. (b) *Secure C-Element* without internal Z states.

4.2.2.1 Transistor structure for the NAND synchronization part

The four C-Elements can be implemented by the memory-less C-Elements of Fig. 4.3. This C-Element (whose reset is omitted) provides two security features:

1. Like the so-called *symmetrical C-Element* [93], its power consumption is same whether an input arrives on port A or B (condition 1 is fulfilled.)
2. When it is used, like in Fig. 4.1, as an input decoder, it can receive two consecutive events on the same input. After the two events have arrived, the C-Element returns in the same state as it was before, leaving no evidence that either A or B has received two consecutive events. This results from the fact that, in the NULL state ($A=B=0$), all the nodes are driven.

4.2.2.2 Transistor structure for the NAND computation part

The two 3ORs of Fig. 4.1 cannot be implemented by those of Fig. 4.2, since they violate the first condition discussed in Sec. 4.2.1.2. As a matter of fact, the electrical consumption of the 3NOR of Fig. 4.2 is different depending on which port (A, B or C) an input arrives. A systematic way to get rid of the difference of the 3OR transistor structure seen from either input consists in implementing all the 3! 3OR gates with the inputs permuted and to short together their outputs, as depicted in Fig. 4.4.

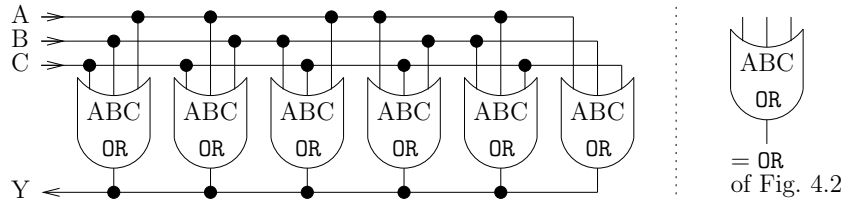


Figure 4.4: Implementation of a 3OR without electrical symptom, built out of 3! Z-free 3ORs.

A lighter solution consists in assembling 2ORs to build the 3OR. Using only two 2ORs does not allow to build a gate whose 3 inputs are equivalent, as shown in Fig. 4.5. However, 3 2ORs (Fig. 4.6), one being dummy, can be assembled in such a way it is impossible to tell, from the power consumption analysis, which input has been activated from 0 to 1.

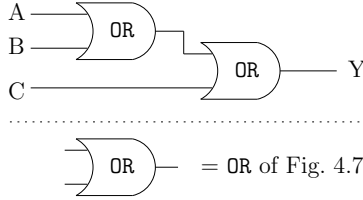


Figure 4.5: 3OR gate whose inputs are not equivalent.

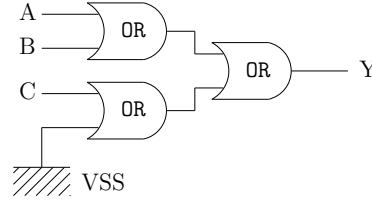


Figure 4.6: 3OR gate whose inputs are equivalent.

In CMOS logic, a 2OR gate is typically build with a 2NOR followed by an INV. A 2NOR gate suitable for the implementation of a 3OR gate (as depicted in Fig. 4.6) is shown in Fig. 4.7. It

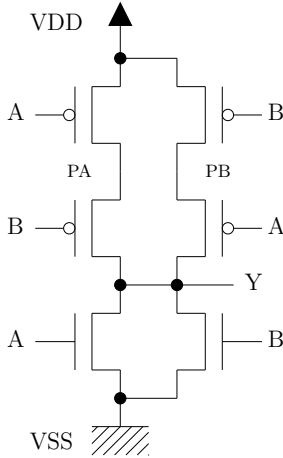


Figure 4.7: Balanced 2NOR, without un-driven nodes provided $A \neq 1$ and $B \neq 1$.

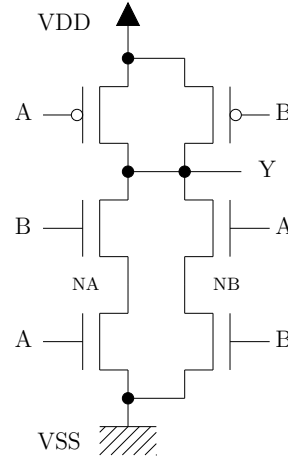


Figure 4.8: Balanced 2NAND, without un-driven nodes provided $A \neq 0$ and $B \neq 0$.

is symmetric with respect to the input exchange $A \leftrightarrow B$ but do contain Z states in its truth table. However, for every computation, at most one among the three inputs has a transition $0 \rightarrow 1 \rightarrow 0$, depending whether y_0 or y_1 is set to 1 by the computation (*cf.* Fig. 4.1). Now, in the activated 3OR gate, the inputs of one first-level 2OR (or actually 2NOR) and of the last-level 2OR turn from $(0, 0)$ to $(0, 1)$, or vice-versa, while the second first-level 2NOR inputs are left unchanged at $(0, 0)$. However, in the transition $(0, 0) \rightarrow (0, 1) \rightarrow (0, 0)$, all the nodes of the 2OR are driven. As a consequence, the final state of the 2NOR gate is the same as it was before the computation.

It would not have been so if we had needed a 2NAND (*cf.* Fig. 4.8). The 2NAND memorizes “VSS” in its node NB after a sequence $(A, B) : (0, 0) \rightarrow (0, 1) \rightarrow (0, 0)$. Then, in the next computation,

- if the same sequence repeats, NB remains at potential VSS, and no power is dissipated,
- if the opposite sequence (namely $(0, 0) \rightarrow (1, 0) \rightarrow (0, 0)$) happens, node NB is charged to

“VDD” while NA discharges. Additional power is thus consumed, revealing a syndrome on the gate input data.

Moreover, the same problem would happen on the 2NOR of Fig. 4.7 if the NULL state is (1, 1) instead of (0, 0). In this case, the PA and PB node of the 2NOR would be memorizing, allowing for deductions on the data. The RTZ protocol (scheme (4.1)) is thus the one possible alternative compatible with a balanced and memoryless computation part.

4.2.3 Use in a regular design flow

The symmetrical design of the C-Elements (Fig. 4.3) and of the 2NOR (Fig. 4.7) gates needed to build the RTZ dual-rail NAND gate (Fig. 4.1) must be kept in the layout. A possible symmetrical layout for the C-Element is given in Fig. 4.9. The layout is given here in a so-called *scalable CMOS* (SCMOS) technology from MOSIS [4]. In this technology, all the design rules are expressed as a multiple of a parameter λ . For instance, the minimum gate width is given by $2 \times \lambda$; thus in a 130 nm technology, $\lambda = 65$ nm. As in our case the target technology is 130 nm or beyond, the rules are in fact the deep sub-micron variant of SCMOC, called SCMOS_DEEP. The layout editor is GNU/Electric [1].

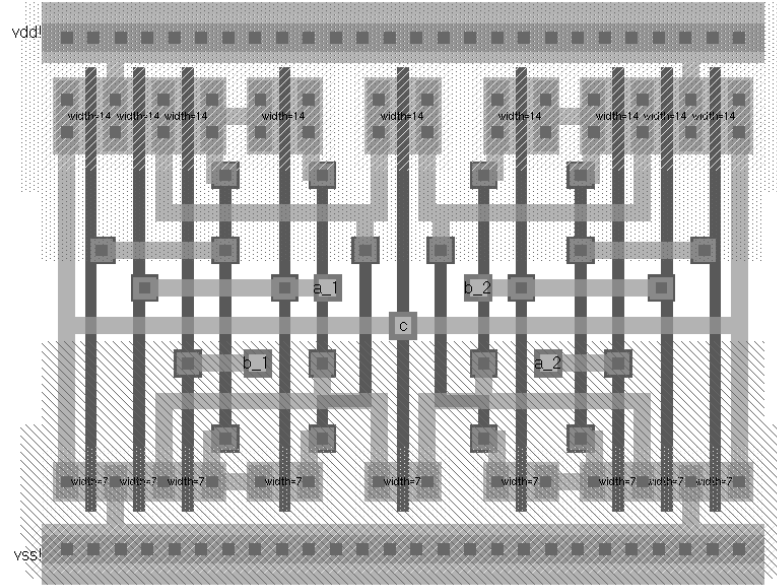


Figure 4.9: Symmetrical layout of the secured C-Element of Fig. 4.3.

Moreover, the dual-rail wires must be routed together.

The secured gates being delay-insensitive, they can be used in an asynchronous logic block [64].

4.2.4 Performances

Spice simulations of the power consumption of the secured dual-rail RTZ NAND gate are shown in Fig. 4.10 (a). There is a slight difference between the traces corresponding to two computations that lead to a different output (y_0 or y_1), as shown in Fig. 4.10 (b). The difference, null in average, arises from the fact that an internal node shortened to ground by a wire is not equivalent to a node connected to ground by a closed transistor (*cf.* dummy 3OR of Fig. 4.1).

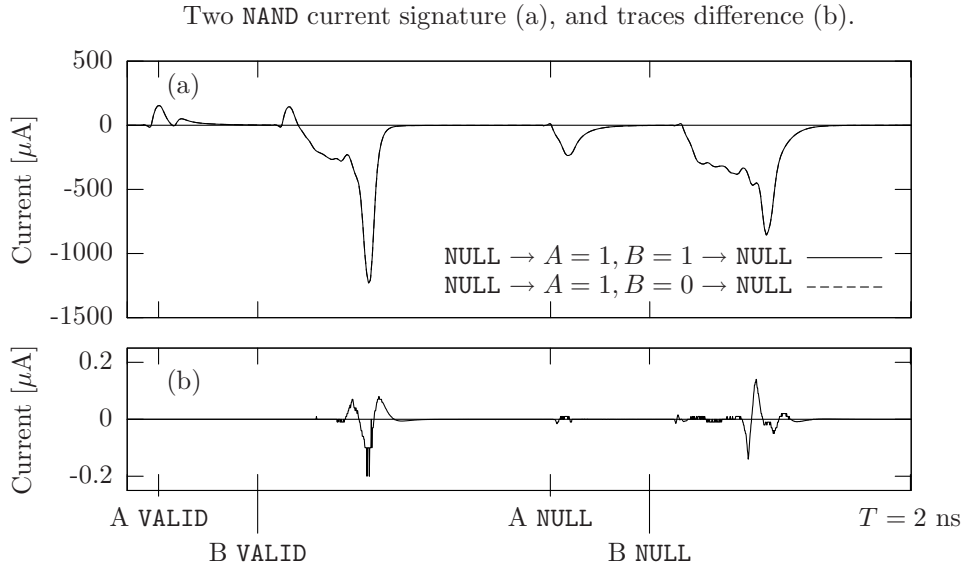


Figure 4.10: Secured NAND current signature. In (a), a transition from $A = 1$ and $B \in \{0, 1\}$ to the NULL state is shown. In (b), the difference according to the target value of B is highlighted.

Thus, at least in simulation (albeit without taking into account of the technological dissipation with a Monte-Carlo analysis), the NAND gate is definitely robust against power leaks.

Tab. 4.1 summarizes the performances of the regular and secured NAND gate.

The NAND gate is large, but the conditions for antennæ effects to manifest are never met. They consist in a possible CMOS transistor gate damage during the fabrication process whenever the surface of the metal exposed to a plasma is greater than a certain ratio of the area of the gates. In the targeted 130 nm technology, the minimum cumulated ratio equals 600, which remains large.

Table 4.1: Standard and secured NAND gate performances confrontation.

Performance	Standard cell	Secured cell
Surface	4 transistors	112 transistors
Propagation $0 \rightarrow 1$	0.03 ns	$2 \times 0.19 \text{ ns}$
Propagation $1 \rightarrow 0$	0.02 ns	$2 \times 0.19 \text{ ns}$
$10 \log(P_{\uparrow}/P_{\downarrow})$	23.2 dB	$7.2 \cdot 10^{-4} \text{ dB}$

4.2.5 The “SecLib” library

Based on the example of the secured NAND gate, a complete library named “SecLib” is built. The cells are called “S<f>-X<d>”, where:

- the logical function is <f> and
- the driving capability is <d>.

Table 4.2: Taxonomy for the 2-input cells $(A, B) \mapsto \langle f \rangle(A, B)$ of SecLib.

Truth Table				Logical Function	$\langle f \rangle$
00	01	10	11		
0	0	0	0	0	-
0	0	0	1	$A \cdot B$	AN2
0	0	1	0	$\overline{A} + B$	NR2A
0	0	1	1	A	-
0	1	0	0	$A + \overline{B}$	NR2B
0	1	0	1	B	-
0	1	1	0	$A \oplus B$	E02
0	1	1	1	$A + B$	OR2
1	0	0	0	$\overline{A + B}$	NR2
1	0	0	1	$\overline{A \oplus B}$	EN2
1	0	1	0	$\overline{B}$	-
1	0	1	1	$\overline{A \cdot B}$	ND2A
1	1	0	0	$\overline{A}$	-
1	1	0	1	$\overline{A \cdot \overline{B}}$	ND2B
1	1	1	0	$\overline{A \cdot B}$	ND2
1	1	1	1	1	-

The taxonomy of the functionality of the non-trivial two-input gates available in SecLib is given in Tab. 4.2. The following name mangling is used:

1. The prefix indicates the Boolean function class, as specified in this hash table:
(AN2 $\rightarrow$ “and”, EN2 $\rightarrow$ “xnor”, E02 $\rightarrow$ “xor”, ND2 $\rightarrow$ “nand”, NR2 $\rightarrow$ “nor”, OR2 $\rightarrow$ “or”).
2. The name can be postfixed with one of the inputs, conventionally called A and B . It signifies the inversion of the given input prior to evaluating $\langle f \rangle$.

Notice that because of the de Morgan law, NR2A could also have been called AN2B, *etc.*

In a view to ease the library maintainability, the layouts are hierarchical. The approach consists in reusing as many components as possible. The result is a library based on specialized templates, depicted in Fig. 4.11. The specialization is implemented with the punctual addition of vias or metal-2 bridges.

The secured NAND gate layout, called SecLib:SND2, is represented in Fig. 4.12.

An asymmetrical multiplexer is added to the library. This cell waits only on the selected input. It can be used for the interface, because the arrival date of an external data is a public information. In a cryptoprocessor, its goal is to inject the RTZ dynamic from the outside. This cell is called SMUX2 and is represented in Fig. 4.13.

SecLib is also comprised of non functional gates. The full content of SecLib is listed below:

- **Scannable Flip-Flops**, with or without initialization
- **Buffers**: driving strength $\in \{\frac{1}{2}, 1, 2, 4, 6, 8, 10, 12\}$
- **Logical Inverter**: dual-rail swap $(y_0, y_1) = (a_1, a_0)$

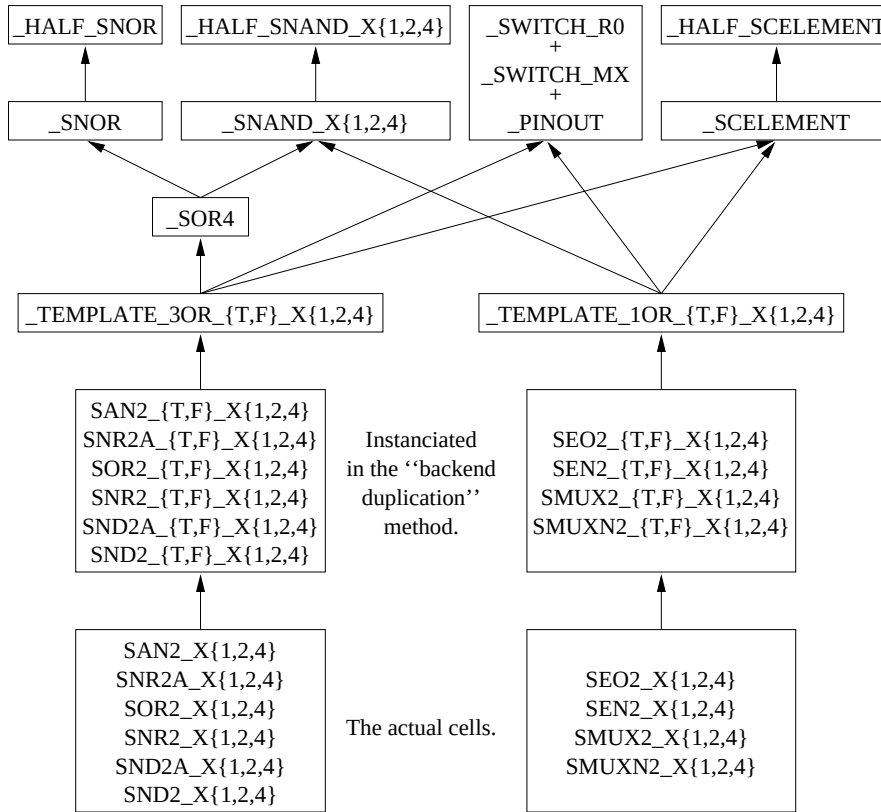
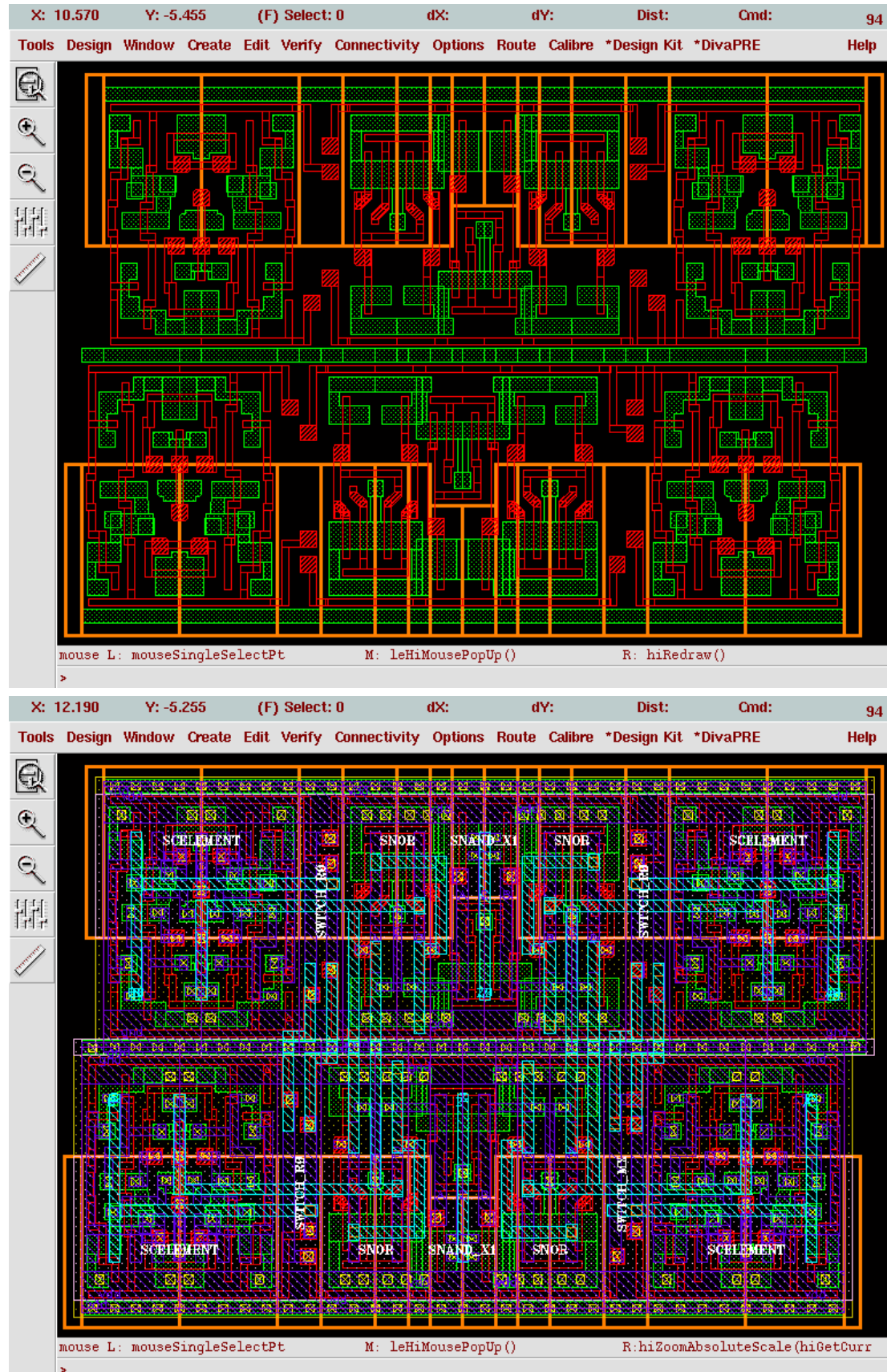
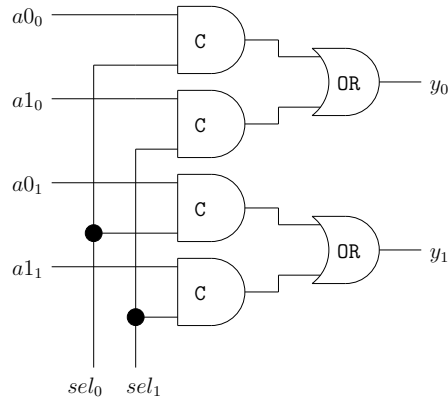


Figure 4.11: SecLib layout database organization.

Figure 4.12: Example of the layout of SND2: the transistors (*upper*), the complete cell (*lower*.)

Figure 4.13: The asymmetrical $2 \rightarrow 1$ multiplexor SMUX2.

- **Two-input Secured Cells** [42], also known as QDI (Quasi Delay Insensitive) gates in the asynchronous logic community (see Tab. 4.2.)
- **Asymmetrical $2 \rightarrow 1$ multiplexor**: for data input into the cryptographic core pipeline

An overview of SecLib organization in the Cadence DESIGN FRAMEWORK II library manager is provided in Fig. 4.14.

The SecLib library can be applied to reconfigurable circuits. In these circuits, the elementary computing elements (the “Logic Element”) is a programmable versatile cell. For this reason, it is larger than a standard cell. In a view to designing an FPGA robust *by design*, the SecLib logic is a suitable candidate: the area overhead intrinsic to this logic is reduced by the need of versatility. A schematic of a two-input reconfigurable secured gate is given in Fig. 4.15. This gate is able to compute arbitrary expressions in disjunctive normal form (DNF) $y(a, b) = \sum_{(i,j) \in \{0,1\}^2} (\bar{i} \oplus a) \cdot (\bar{j} \oplus b) \cdot f_{ij}$, using dual-rail RTZ logic. Notice that the reconfigurable capability of this cells enables advanced strategies for resilience. The “*on field*” agility indeed allows a dynamic adaptation whenever necessary.

4.2.6 Interconnect involvement in a circuit security

The amplitude of a single net power signature is proportional to the capacitance it drives. A first estimation of the capacitances is given by the synthesizer, based on *wire-load tables*. A deeper analysis of the capacitances requires the place-and-route (P&R) step of the synthesized netlist. The process is shown in Fig. 4.16 and commented below:

- ❶ The source code, written in an hardware RTL language such as VHDL, can be simulated and synthesized
- ❷ Thanks to a logical synthesizer, a netlist of gates mapped into a given technology is produced. The non-placed-nor-routed netlist is designated by the extension `*.vm`
- ❸ A place-and-route tool allows to obtain the final netlist, of extension `*.vo`, along the extracted physical information, in the standard parasitics exchange format (SPEF [7, Sec. 9 pp. 288–332], of extension `*.spef`).

The state-of-the-art Cadence RTL to silicon tools divide into a synthesizer (`pks_shell` family) and a place-and-route tool (`encounter`.) However, both tools are built upon the same

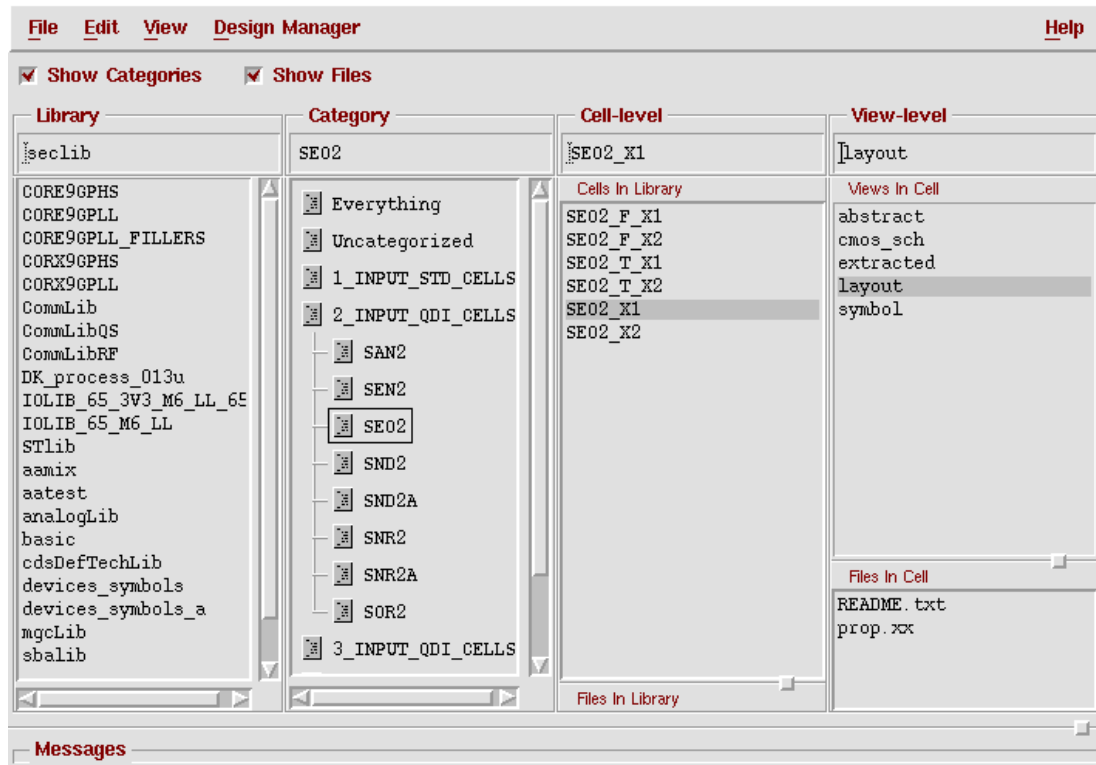


Figure 4.14: SecLib viewed from the Cadence® dfII Library Manager.

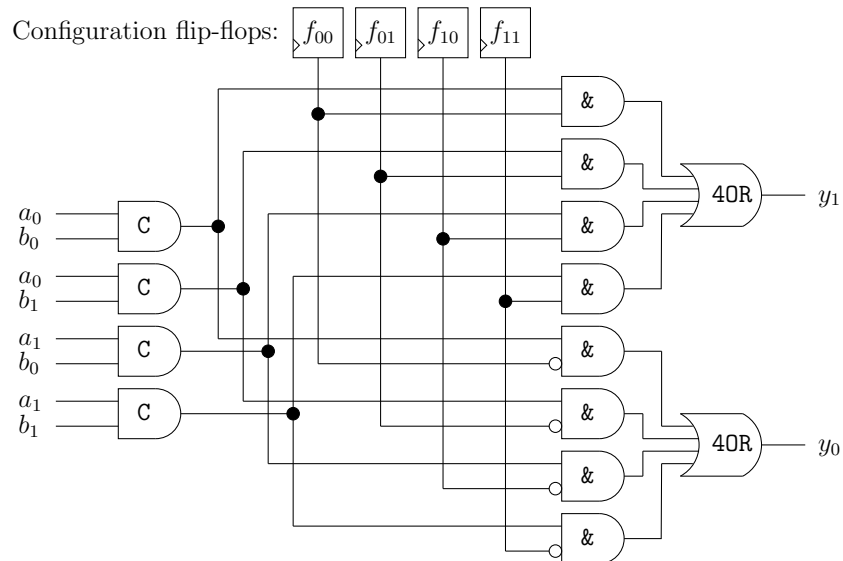


Figure 4.15: Example of a reconfigurable 2-input DI secured gate.

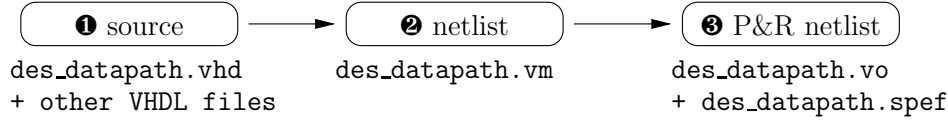


Figure 4.16: Design flow steps from the RTL source code (in VHDL, *.vhd files), via the netlist (in Verilog, *.vm files), to the placed-and-routed netlist (in Verilog, *.vo files) along with the extracted physical information (in SPEF, *.spef files.)

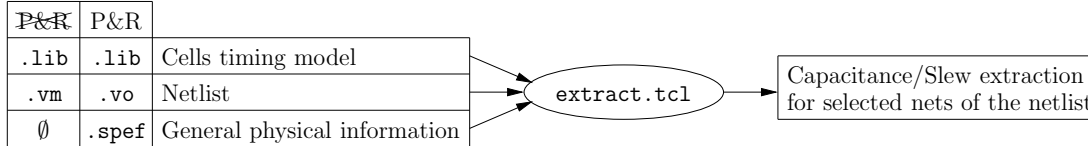


Figure 4.17: Physical quantities extraction from the netlists, with (.vo) or without P&R information (.vm.)

timing computation software library, called CTE (Common Timing Engine, described in [23].) For this reason, the netlists, be they placed-and-routed or not, can be reinjected in the synthesizer to extract quantities such as:

- net total capacitances and
- slew rates (transition durations.)

This process is illustrated in Fig. 4.17. It boils down to a simple TCL script for the synthesizer Cadence `pks_shell`, illustrated in Fig. 4.18 with a technology `${lib}`, a P&R netlist `${vo}` and the associated parasitics `${spef}`.

The extracted capacitances can be split into two parts:

1. the gate capacitance (linked to the standard cells) and
2. the interconnect capacitance (linked to the routing.)

The capacitances of the R register from the iterative DES architecture presented in Sec. 3.5.3.1 is studied in this section. Three capacitances values are analyzed:

1. an estimation of the capacitances after synthesis,
2. an extraction of the capacitances after P&R with an out-dated three-dimensional field-solver (version 3.2 of First Encounter [47], *aka* FE 3.2),
3. an extraction of the capacitances after P&R with an up-to-dated field-solver (FE 4.1.)

```

read_dotlib          ${lib}; # Cells definition and timing
read_verilog -structural ${vo}; # The P&R design netlist
read_spef            ${spef}; # The P&R parasitics information
# Use "report_net" to extract the physical quantities that are relevant
  
```

Figure 4.18: The “extract.tcl” script mentioned in Fig. 4.17.

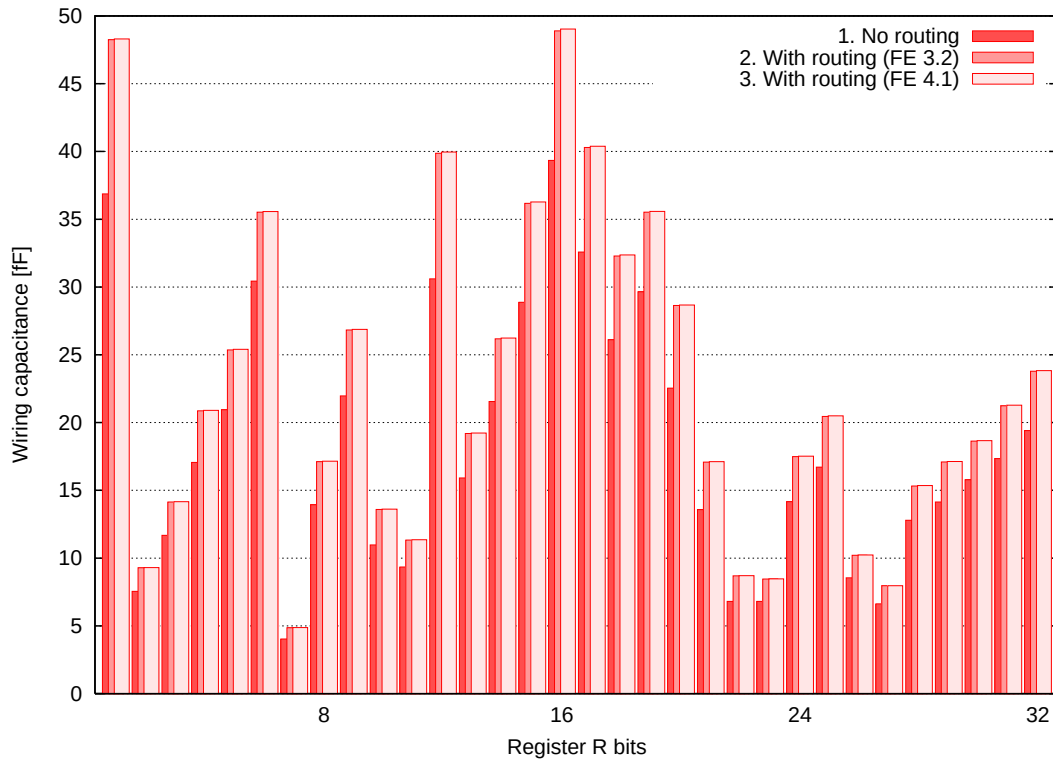


Figure 4.19: Absolute value of the interconnect capacitances driven by nets R[1:32] nets.

We find that the gate capacitances are equal for the three analyses. A comparison between the interconnect capacitances is reported in Fig. 4.19.

The deviation from the out-dated version 3.2 w.r.t the current version 4.1 is equal to 0.22 %, which is negligible. This means that CAD tools are reliable. However, the capacitances estimated after synthesis differs much from the extracted values after extraction (22 %). It can nonetheless be noticed that the estimation provided by the synthesizer is always pessimistic. If the synthesizer’s predictions are magnified by a factor of 22 %, then the deviation shrinks to 4.0 %. The conclusion is that *relative* capacitance values can be computed from the sole synthesis results with an accuracy of a few percents.

In the rest of the document, the most accurate estimation (namely version 4.1 of the “First Encounter” P&R tool) is taken as the reference. As shown in Fig. 4.20, the wiring contribution to the capacitances is in average 53 % of the total capacitance. As a consequence, from 130 nm technology and beyond, routing capacitances become larger than gate capacitances, at least for nets with fan-outs of 6 or 7 (which is the case for register R.) It is thus mandatory to provide a secured routing method. This is the topic of the next section 4.3.

4.3 The “backend duplication” method: a place-and-route strategy for secured ASICs

Several types of logic gates suitable for leakage-proof computations have been put forward [103, 105, 64, 42, 80]. This section describes a method, called “backend duplication” [102] to assemble secured gates into leakage-proof cryptoprocessors. The section addresses all the aspects involved

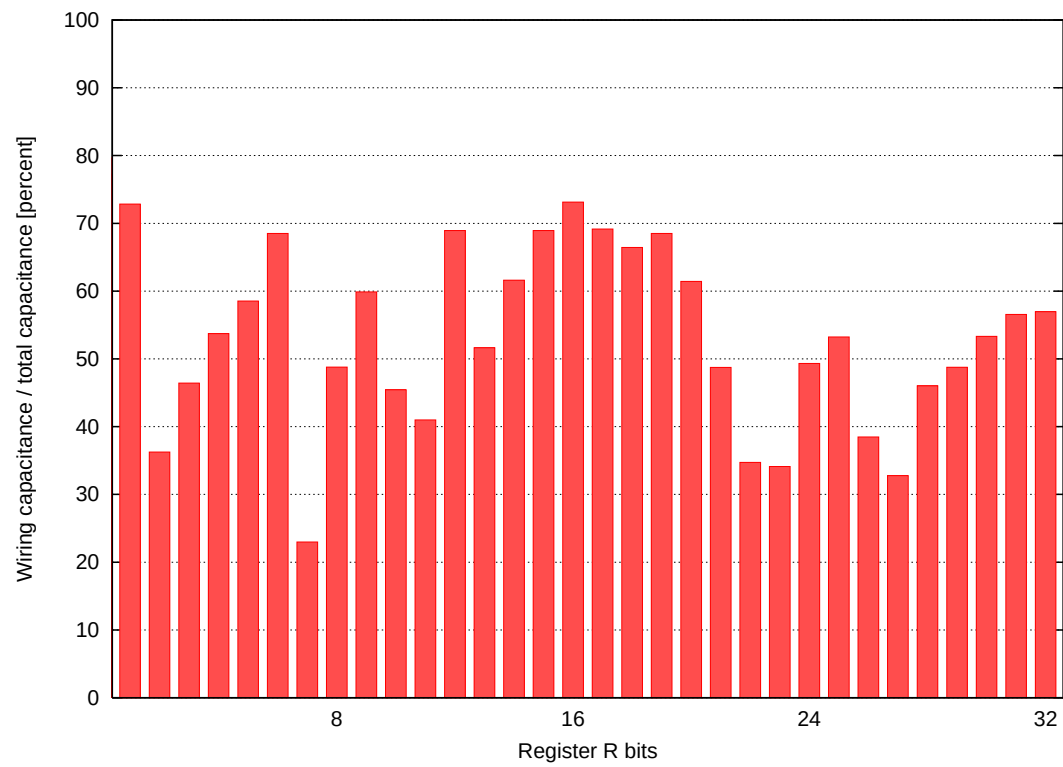


Figure 4.20: Ratio between the interconnect and the total capacitance of R[1:32] nets.

in the backend design of secured hardware. The “backend duplication” method achieves the place-and-route of differential netlists. It allows for 100 % placement density and for balanced routing of dual-rail signals. Wires of every other metal layer are free to make turns. In addition, the method does not require any modification to the design rules passed to the router. The “backend duplication” method has been implemented in 0.13 μm ASIC technology and successfully tested on various ciphers. The example of the design of a DES module resistant against side-channel attacks is described into details.

4.3.1 Using differential logic to thwart SCA

It has been shown that sensitive information can be extracted from cryptographic hardware either by spying physical quantities or by injecting faults. The first type of attack is often referred to as “side-channel attack” (SCA [53, 54, 37]), whereas the second one is also known as “fault attack” (FA). Two classes of countermeasures against SCA have been put forward. The first idea is to shield the hardware at the algorithmic level: the data manipulated by the cryptoprocessor is masked or protected by secret-sharing methods. The second idea is to build the hardware using only leakage-proof gates, so as make sure that the overall cryptoprocessor is, in turn, leakage-proof.

This section focuses on the implementation of the latter class of countermeasures. Many leakage-proof logic styles have been published. The level of protection the secured gates provide depends upon their specification:

1. SABL [103] is a logic consuming a nearly constant current.
2. WDDL [105] uses dual gates pairs to ensure a constant activity, although the power consumed by each gate of the pair is not the same.
3. Speed-independent (SI) logic presented in [64] features a consumption independent on the input data configuration. It also shields against the leakage of the signal transitions timing by synchronizing the inputs.
4. Refinements [42] of the previous solution also ensure that parasitic capacitances are unconditionally unloaded between two computations.

Some of those methods, for instance methods 3 and 4 above (nicknamed “SecLib” in the rest of this section) can also embed an error-detection feature. The mechanism, based on an alarm propagation, is explained in [64]. Nevertheless, resistance to faults injection is not covered in this section.

The *logical* part (coding, functionality verification, refinements for synthesis) in a design targeting FPGA or ASIC implementation is called *frontend*. The *physical* part (mainly consisting in place-and-route, but extensive description is provided in Sect. 4.3.2) is called *backend*. The common point to the secure gates listed above is the use of differential logic with a 4-phase protocol, such as “return to zero” (RTZ) or any variation [96]. It has already been stressed that the security of individual gates can extend to a netlist of gates only provided that the interconnect is kept differential [106]. Nonetheless, most articles evade the question of the implementation of a secure backend design using the gates mentioned above.

Given the complexity of backend flows in sub-micron technologies, a simple way to realize the secure backend is necessary. We provide in this article a method, called “backend duplication”, that integrates the secure place-and-route into any preexisting backend flow without modifying the design rules.

The rest of the article is organized as follows: the “backend duplication” is presented in Sect. 4.3.2. The method is applied to some secured gates primitives in Sect. 4.3.6. A case study, namely a DES cryptoprocessor, is provided in Sect. 4.3.8. This example was actually fabricated in HCMOS9GP 0.13 μm technology from STMicroelectronics using the method presented in

this section. This section contains an evaluation of the cost and of the security increase provided by the use of the “backend duplication”. Finally, Sect. 4.3.9 concludes this section.

4.3.2 The “backend duplication” method

4.3.2.1 Regular “place-and-route” ASIC design flow

In a standard cell flow, cryptographic functions are synthesized into a netlist of primitive gates. Then, the gates are placed into rows (see Fig. 4.21(a)). In each row, the gates are abutted, so that they share the ground (VSS) and the power (VDD) lines. When two gates are not placed side by side, a “filler” cell can be added in-between to ensure the continuity of the supply lines. In sub-micron technologies, there are enough levels of metal to allow the routing of the interconnect over the standard cell rows. For this reason, the rows are themselves abutted. Thus, the supply lines are shared between adjacent rows. This is achieved by flipping upside-down every other row: the ground (resp. the power) of one row is merged with the ground (resp. the power) of the lower (resp. the upper) row, (see Fig. 4.21(b)).

Sub-micron technologies allow for 45 degree wire routes, but this feature is not yet implemented in commercial routers: currently, the routing is still Manhattan. Moreover, the most popular routers are also grid-based. Metal wires are only instantiated along a virtual routing tracks superimposed on the floorplan (see Fig. 4.21(c)). It is thus customary to attribute a preferred direction to every routing layer. However, routers consider the preferred direction only as a recommendation. The convention we use in this section is that odd metal levels (metal 1, metal 3, and so forth) are preferentially routed vertically, whereas even metal levels are preferentially routed horizontally.

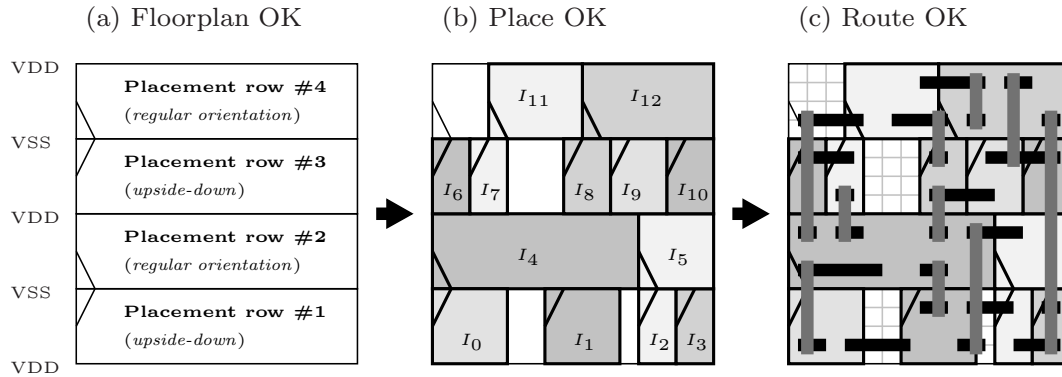


Figure 4.21: Illustration of the regular (and insecure) “Place-and-Route” ASIC design flow.

4.3.2.2 The “backend duplication” method overview

The “backend duplication” addresses the strengthening against SCA of sensitive ASICs (smart-cards, hardwired cryptoprocessors, *etc.*) It consists in a single manipulation of the backend layout to ensure the security of its interconnect. However, this method shall not be confused with the tailored duplication method for software or dedicated hardware implementations [41].

The basic idea of the “backend duplication” method is to apply a regular backend flow on a single-ended (as opposed to duplicated) netlist, taking care to leave enough room on the floorplan for the duplication of the placed-and-routed netlist. The duplication basically demands that every other row be kept free, which is typically achieved by obstructing every other row for placement.

The next aspect concerned with duplication is the interconnect. To make it possible to duplicate the interconnect, the vertical wires, that connect every other row, are forced to occupy only one routing channel in two. This ensures that a simple right shift of the vertical routing by a routing pitch (i.e. the distance separating two routing tracks) does not create electrical shorts. As a consequence, vertical wires must be straight. If they were able to make turns, they would cross the adjacent routing tracks that are kept free for the duplicated vertical routes. On the contrary, wires of the “horizontal routing layers” are left free to make turns, as long as they remain in their placement row. Indeed, if the horizontal routing is confined within one row over two, the duplication of the “horizontal” wires in the upper or the lower rows does not interfere with the wires in the current row.

The constraints imposed to the place-and-route tool summarize as follows: as the design must be translated vertically by the height of one placement row (`ROW_HEIGHT`) for placement reasons and horizontally by one routing pitch (`PITCH`) for routing reasons, the whole placed-and-routed design is scheduled to move by a $(\delta x, \delta y) = (\text{PITCH}, \text{ROW_HEIGHT})$ vector translation. In backend taxonomy, this translation actually coincides with the minimum “placement site”.

At that point, the result of the duplication is two identical netlists interleaved one into the other. Notice that the netlists cannot be “de-interleaved” because they are not independent: some signals must be exchanged locally between abutted gates. As we will see in Sect. 4.3.6, it happens for the inverter gate in SABL and WDDL (Tab. 4.3(b)) and for all gates in SecLib (Fig. 4.29).

The chip finishing steps shall not delete the indistinguishability of the two netlists. For instance, the dummies generator must be constrained to add dummies (metal pieces added randomly to fulfill the minimum density design rules) only in the rows in which placement is allowed. Afterwards, dummies are duplicated and translated by a placement site: they end up in the same routing environment as initially (no short is created) since the routing was duplicated in the same manner.

4.3.3 The constraints required by the “backend duplication” method

As mentioned above, the “backend duplication” method is implemented in two stages, consisting in:

- (1) constraining the design and
- (2) duplicating the placed-and-routed design.

The constraints can be generated automatically by a script setting the following obstructions:

- **placement blockages** one row over two and on the rightmost placement site of the placeable row,
- **routing blockages** of one track channel over two for vertical metals and over the rows already marked obstructed for standard cell placement for horizontal metals.

Figure 4.30(a) illustrates these constraints on a 16×2 -site piece of floorplan. As far as the routing is concerned, these constraints are more flexible than the ones proposed in the “fat wire” method [106], since only vertical wires are forced to remain strictly straight. The metals whose preferred routing direction is horizontal are free to zigzag, provided they stay within their row. This degree of freedom is not negligible, since there are typically around 12 routing channels per row. This allows for both a more successful and a faster routing.

It must be made clear that the constraints to set for the P&R steps are not a burden from the tool’s point of view. The expression of the constraints only slightly increase the workload of the designer. However, the constraints are so simple that their implementation is worth the effort, given their expected security benefits.

4.3.4 “Backend duplication” method insertion into an existing design flow

As seen in Sect. 4.3.3, and contrary to [106], the “backend duplication” method need not redefine the design rules. It only relies on constraints on the CAD software. A typical backend flow includes the steps shown in Fig. 4.22. The insertion of the “backend duplication” consists in adding three steps (*i*, *ii* and *iii*).

Regular backend flow:	Flow compatible with the “backend duplication”. Added steps:
.....←	<i>i</i> : Floorplan dimensioning
- Floorplanning	
.....←	<i>ii</i> : Obstructions implementation
- Place-and-route	
- Clock tree generation	
- Scan chain optimization	
- Antenna effects correction	
- Custom steps, like ECO or SI fix	
- Dummies placement	
.....←	<i>iii</i> : Duplication

Figure 4.22: Typical backend flow and modifications (steps *i*, *ii* and *iii*) to implement the “backend duplication” method.

***i*. Floorplan dimensioning.** As a matter of fact, the floorplan of an design block is made up of two parts: the *core*, devoted to the standard cells placement and the *die*, that covers the core and an extra channel surrounding it. It is used for example to route a supply ring. The core horizontal dimension must be an even number of the routing PITCH and the vertical dimension an even number of ROW_HEIGHT. This condition ensures that the placement and the routing within the core do not extend out of the core after duplication.

The core can either be checked and repaired if one of the figures is odd or generated automatically. To end up with a core of density d and of aspect ratio r , the first step is to generate a core of density $d/2$ and of aspect ratio $r/2$ before duplication. Then the core dimensions (x, y) are retrieved, and a new core with the dimensions:

$$x' = \left\lceil \frac{x}{2 \times \text{PITCH}} \right\rceil \times 2 \times \text{PITCH}, \quad y' = \left\lceil \frac{y}{2 \times \text{ROW_HEIGHT}} \right\rceil \times 2 \times \text{ROW_HEIGHT}$$

is regenerated. Its density is slightly less than d and its aspect ratio roughly equal to r .

***ii*. Obstructions instantiation.** The constraint script described previously in Sect. 4.3.3 can be generated automatically as soon as the floorplan dimensions are known. This script is sourced after floorplanning and before place-and-route.

***iii*. Duplication.** As far as standard cells are concerned, the duplication consists in a translation by a placement site followed by an horizontal flipping of each row.

The routing duplication is a bit more complex than a mere translation. Indeed, the design pins extend over the core to reach the die boundary. If the routing was simply translated, the duplicated design would have pins both inside and outside the die. To avoid this shortcoming, the routing extremities (u, v) of every wire undergo this transformation:

- if (u, v) belongs to the core, then $(u', v') = (u + \text{PITCH}, v + \text{ROW_HEIGHT})$,

- otherwise $(u', v') = (u, v)$.

Additionally, to prevent shorts, the constraints described in Sect. 4.3.3 actually extend till to die limits and the routing channels that are entirely outside the core are obstructed. These transformations are illustrated on Fig. 4.30(b).

The information needed to apply the duplication is the orientation and position of standard cells and the routing coordinates. The design exchange format (DEF) typically contains all this information. Given the simplicity of the DEF syntax and the availability of parsers [56], the duplication can be implemented easily.

It is also a good idea to apply the duplication on the Verilog netlist: it consists in duplicating all wires and all leaf instances (i.e. standard cells). Verilog parsers are easy to write, even from scratch. The key benefit of generating the duplicated Verilog netlist is to enable LVS verification.

4.3.5 Comparison of the “backend duplication” method with related works

K. Tiri [104] noticed that the balancedness of the routing is crucial to effectively protect a differential circuit against SCA. The solution put forward in [106] is based on “fat wires” routing: a large wire is first routed and then split into two minimum-sized wires. This method implies that:

- Specific design rules must be written for the “fat wires”.
- The only way for a wire to turn is to change layers.
- For the “fat wire” to access the pins of standard cells, their layout must be redefined.

The “backend duplication” implies none of these assumptions.

The experimental DPA [54] of F.G. Bouesse *et al.* [21] also showed that the weakest nodes in a differential layout correspond to unbalanced pairs. The backend correction flow described in [20] is iterative: the design is successively routed and analyzed, until every dual-rail pair is balanced. The analysis consists in the collection for every node of the sum of the parasitic elements extracted after every routing (more details in Sect. 4.3.8.2). This method requires a complex strategy to constrain the router and a non trivial algorithm to guide the iterative process towards a convergence point. On the contrary, the routing generated by “backend duplication” is balanced by design. However, the “backend duplication” only handles pairs of signals, whereas the iterative method [20] can route both dual and single-rail signals (data is dual-rail; acknowledge is single-ended.)

4.3.6 Suitability of the “backend duplication” method with some logic styles

4.3.6.1 Backend duplication for WDDL

The wave dynamic differential logic (WDDL, [105]) is a design style that uses standard cells by pairs, in such a way that at any step of the computation, one and only one of the two gates has a transition. This behavior masks the fluctuations of the power consumption due to irregular activity: the activity of a WDDL circuit is constant. The computations are split into successive *precharge* and *evaluation* steps. A Boolean function $e_{i \in \{0,1,\dots\}} \mapsto f(e_i)$ is computed using the two dual gates $f_T(e_i)$ and $f_F(e_i)$ that satisfy:

$$\begin{cases} \text{During precharge:} & \exists i, \quad f_T(e_i) = f_F(e_i), \\ \text{During evaluation:} & \forall i, \quad f_T(e_i) = f_F(\bar{e}_i). \end{cases} \quad (4.3)$$

Table 4.3: Duality: definition, examples (a) and WDDL identity for the inverter (b).

(a)	Regular gate	Dual gate
Definition	$f(e_i)$	$\overline{f(\overline{e_i})}$
Examples	NOT	NOT
	NAND	NOR
	$\Pi \Sigma e_i$	$\Sigma \Pi e_i$

(b) Dual inverter

Regular inverter

Table 4.4: Truth table of the two dual functions NAND / NOR.

		NAND			NOR
e_0	e_1	$\overline{e_0 \cdot e_1}$			$\overline{e_0 + e_1}$
0	0	1			1
0	1	1			0
1	0	1			0
1	1	0			0

Table 4.3(a) provides some examples of dual gates pairs suitable for WDDL. If the condition on the precharge in (4.3) cannot be met, the identity shown in Tab. 4.3(b) solves the problem out. The truth table of two dual gates (refer to Table. 4.4) shows a symmetry, that can also be observed at the transistor level, as shown in Table. 4.5.

The symmetry illustrated in Table. 4.5 suggests that standard cells are ready to be used in a WDDL flow using the “backend duplication” method. This is actually only partially true: the structures in transistors indeed perfectly superimpose, but in practice, PMOS (*symbol*:) are drawn wider than NMOS (*symbol*:). For this reason, in a commercial standard cell library, the pins of a gate (regular orientation: or R0) and of the X-symmetric (orientation: or MX) of its dual do not match exactly. Nevertheless, as they are located on the routing grid, they usually overlap.

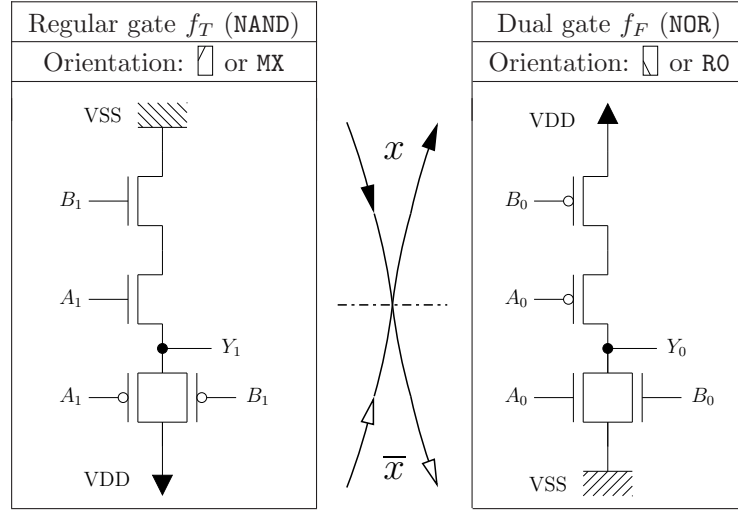
Fortunately, it is easy to work around this difficulty. The procedure begins with an enlargement of the pins. Then, the pins are merged considering the intersection of the enlarged pins. The routing obstructions are basically made up of the metal not included in the union of the newly created pins:

$$\begin{cases} \text{PIN} &= \text{PIN}(\text{NAND}) \cap \text{PIN}(\text{NOR}), & (\text{ } \square \text{ in Fig. 4.27}) \\ \text{OBS} &= (\text{OBS}(\text{NAND}) \cup \text{OBS}(\text{NOR})) \cup (\text{PIN}(\text{NAND}) \triangle \text{PIN}(\text{NOR})). & (\text{ } \blacksquare \text{ in Fig. 4.27}) \end{cases}$$

This procedure can be applied on the sole *abstract* view of the standards cells. Thus a simple LEF parser [56] can be used turn a standard cell library into a WDDL-compliant library. Instead of describing the parser into details, a graphical example on the NAND/NOR and AND/OR gate couples is shown in Fig. 4.27.

As far as cell placement duplication is concerned, the method presented in step *iii* (refer to Sect. 4.3.4) demands that, in addition to the duplication and the flipping, the gate be replaced by its dual.

Table 4.5: Illustration of the NAND/NOR dual gate couple symmetry $\{f_T, (N, P), \mathbf{MX}\} \leftrightarrow \{f_F, (P, N), \mathbf{R0}\}$.



4.3.6.2 Backend duplication for other logic gates

In order to apply the “backend duplication” method to SABL or SecLib, the gates must be split into two parts: one computing *true* values, the other *false* values.

The splitting is straightforward for SABL, as shown in Fig. 4.28.

As for SecLib, the division is a bit less trivial, but is sane since it forces the symmetry of the transistor schematic to be kept in layout view. The placement of each building block of the cell along with the indication of their orientation is provided in Fig. 4.29.

For both SABL and SecLib, the gate pins must be designed in such a way they are left unchanged in a symmetry $y \leftarrow \text{ROW_HEIGHT} - y$ (or $\mathbf{R0} \leftrightarrow \mathbf{MX}$). This condition ensures that a connection to the pin of a regular gate (placed first) also arrives on a pin of the other half of the gate (placed while duplicating the backend at step *iii*). SecLib gates, whose layout is depicted in Fig. 4.12, meet this condition. Additionally, the routing converges faster if the pins are placed on every other vertical routing track: the pins are better accessed if they are not below a vertical routing obstruction.

The intrinsic dissymmetry of WDDL can be corrected by the use of so-called “enhanced-WDDL” cells are based on the standard 3-input majority cell: $(A, B, C) \mapsto Y = A \cdot B + B \cdot C + C \cdot A$. The schematic of the majority and of two dual “enhanced-WDDL” cells is given in Fig. 4.23. The automatic routing of “enhanced-WDDL” requires the use of constants. The figure 4.24 shows that the backend duplication can effectively cope with constants, thanks to appropriate ties to the global power and ground nets VDD and VSS.

4.3.7 Backend triplication

Incidentally, it is also straightforward to adapt this method to a “backend triplication”. The obstructions would basically be placed on two rows or vertical routing channels out of three. The goal of this method would not be to protect the design against SCA, but rather against FA. For this matter, the registers would also need to embed majority voting and error reporting logic. With this modification, the hardware block is able to detect local errors: this method could be an alternate countermeasure to FA where faults can be modeled as SET or SEU (Single Event Transients or Upsets) [57].

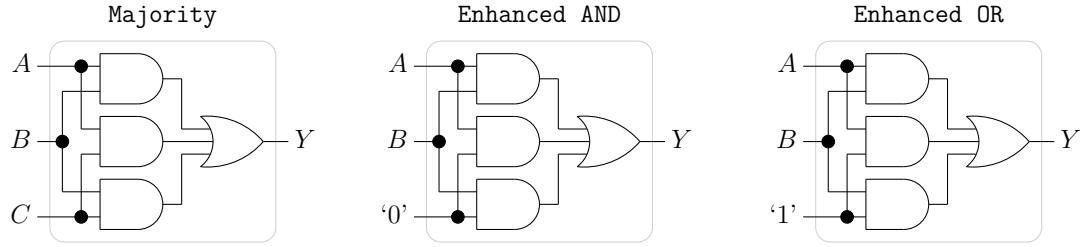


Figure 4.23: The majority standard cell and the two enhanced WDDL cells implementing the AND and OR functionalities.

		constant zero	constant one
<i>false</i>	R0	VDD  '1'	VDD  '0'
<i>true</i>	MX	VDD  '0'	VDD  '1'

Figure 4.24: Realization of constants needed by “enhanced-WDDL” logic.

4.3.8 Implementing a duplicated netlist

4.3.8.1 The example of a secured DES cryptoprocessor design

In this section, we explain how a placed-and-routed netlist obtained by the “backend duplication” method can be embedded into a whole design. First of all, let us notice that after duplication, even global signals are duplicated: the duplicated backend has two clocks and two resets, that must be shorted together. The two scan chains can either be joined or be considered independently.

Most often, the whole cryptoprocessor needs not be secured. The reason is that when implementing a non proprietary algorithm such as DES, the computation steps are public. As a consequence, the control leaks non confidential information. In most designs, the control (algorithm steps) can be clearly dissociated from the datapath (data processing).

It is relevant to derive the control of the duplicated datapath (dual-rail encoding, RTZ protocol) from the original control of the insecure datapath (single-ended, no RTZ): it allows to debug a single-ended control, which is easier to understand and faster to simulate. The method to update the regular control to make it compatible with the duplicated datapath requires that:

- The state machine can be frozen: it has an **enable** input. This enable forces the state machine to work twice as slow as initially to mimic RTZ.
- The control is wrapped by a converter single-to-dual rail for the datapath inputs and dual-to-single rail for its outputs. In addition to converting the control signals exchanged between the datapath and the control, the control wrapper also converts the datapath input and output data. Thus, seen from the outside, the cryptoprocessor keeps a single-ended interface. However, the internal architecture of the datapath is dual-rail RTZ secure logic obtained by “backend duplication”.

When the control is disabled (**enable** = 0), all the input signals of the datapath (provided by the control wrapper) are set to the precharge state (e.g. 00). This solution emulates the

Table 4.6: Timing of backend steps of the regular DES datapath design and an implementation in SecLib style using the “backend duplication” method (block “SDES datapath” of Fig. 4.25).

	Regular design	SecLib design
Placement	1.9 s	6.2 s
Routing	39.0 s	80.0 s
Duplication	-	77.5 s

4.3.8.2 Method cost and security evaluation

The method overhead is assessed below:

- When using WDDL, the circuit frequency is unchanged, but every encryption takes twice more time to execute because of the RTZ protocol. With other logics, the frequency depends on the gates implementation; for instance, SecLib suffers a serious penalty in speed, as shown in Tab. 4.1.
- The area increase of the datapath depends on which secured gates are used. If WDDL gates are chosen, SDES_DATAPATH is simply twice as large as DES_DATAPATH. If SecLib gates are chosen, we obtain a 15 times area increase¹. The overhead of the control area is 14%: the area of the module DES_CONTROL (resp. SDES_CONTROL) is 12 942 μm^2 (resp. 14 788 μm^2 .)

The increase of security can be assessed by the ratio of the two dual lines routing capacitances and resistances. The capacitance “C” accounts for the power dissipation occurring at every transition: $\frac{1}{2} \times C \times (VDD - VSS)^2$. The resistance “R” is responsible for the delay $R \times C$ of the transition propagation. The wire pairs are all the more balanced as the ratios $C(\text{true})/C(\text{false})$ and $R(\text{true})/R(\text{false})$ do not spread much around 1. Figure 4.26 shows the repartition of those ratios for the 2 211 internal wire couples of SDES_DATAPATH. The three data samples correspond to a dual placed-and-routed design, obtained by the “backend duplication” method, a dual placed and regular routed design, and a regular placed-and-routed design. Both the capacitances and resistances were obtained using the RC extractor tool of Cadence SOC/ENCOUNTER. The technological information was produced by the Cadence COYOTE 3D-field solver.

The resistance of a “backend duplicated” circuit against EMA [37] has not been evaluated yet.

4.3.9 Conclusion on the “backend duplication”

Securing a cryptoprocessor against physical attacks (either SCA or FA) can be done at the algorithmic or at the implementation level. This section focuses on the countermeasures on the hardware implementation. Many types of primitive gates suitable for secure computation have been proposed [103, 105, 64, 42, 80, 31, 9], but the issue of building cryptoprocessor out of them is seldom addressed. To the authors’ knowledge, only the “fat wire” method [106] partially tackles this problem since practical aspects of layout finishing are not discussed. In the case of WDDL netlists, the “fat wire” method cannot cope with constants, whereas the “backend duplication” can.

We provide a complete description of a backend flow compatible with all of the above-mentioned gates. The method we describe can apply to all existing flows and requires no modification of the design rules.

¹The SecLib gates were not optimized: a much better ratio can probably be obtained, even without any trade-off on the gate symmetry.

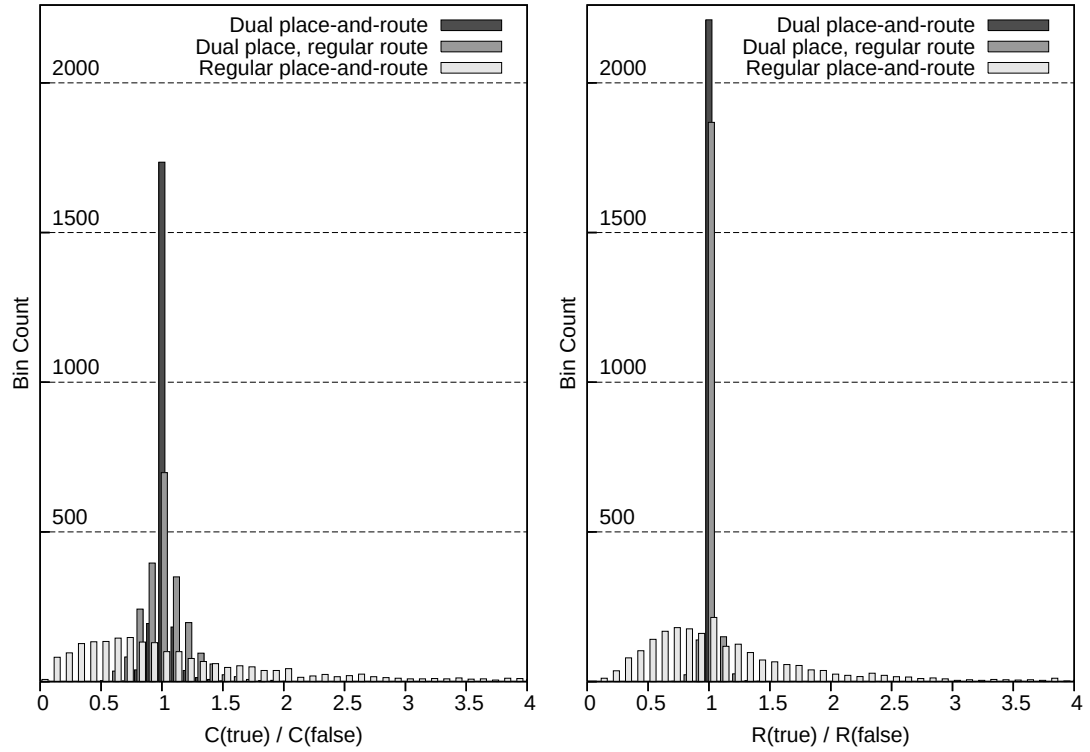


Figure 4.26: Ratio of the capacitances and the resistances of SDES_DATAPATH dual nets.

The “backend duplication” method is illustrated on the example of a DES cryptoprocessor. This example also shows that the method is compatible with a secure partitioning of the design: only the datapath is duplicated. The emphasis is placed on the insertion of the duplicated datapath into the whole DES, whose interface remains unchanged. This case study proves that the hardening of a cryptoprocessor can be fully automated and that the integration of the “backend duplication” method into an existing flow is seamless.

4.3.10 Graphical illustrations of the “backend duplication” method

Figures 4.27, 4.28 and 4.29 show how WDDL, SABL and SecLib gates must be transformed prior to being used in the “backend duplication” design flow.

Figure 4.30 illustrates the “backend duplication” (steps *ii* and *iii*) on a floorplan suitable for the duplication (step *i* was already executed: the floorplan dimensions are even.)

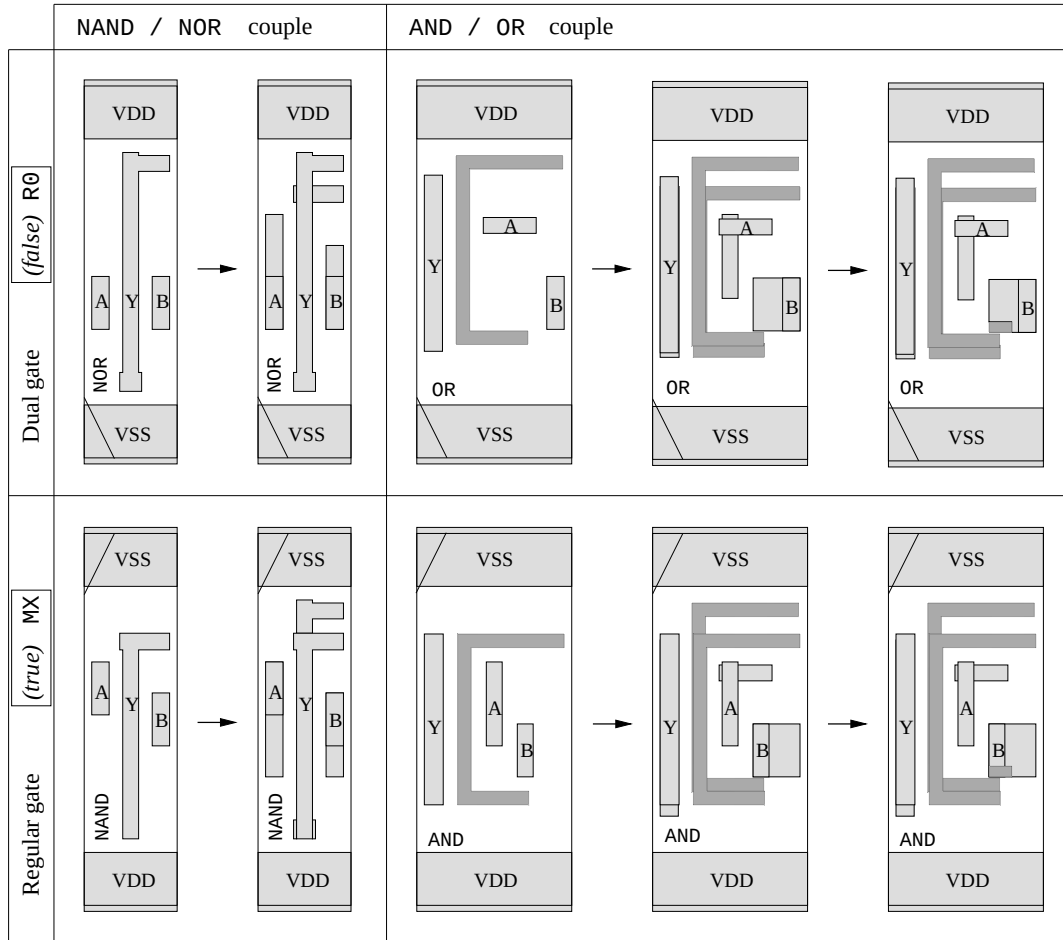


Figure 4.27: Transformation on the *abstracted* views of the standard cells to make them WDDL-compliant [105]. This resulting gate couple satisfies the following condition: the abstract couples $\{f_T, (N, P), MX\}$ and $\{f_F, (P, N), R0\}$ perfectly superimpose.

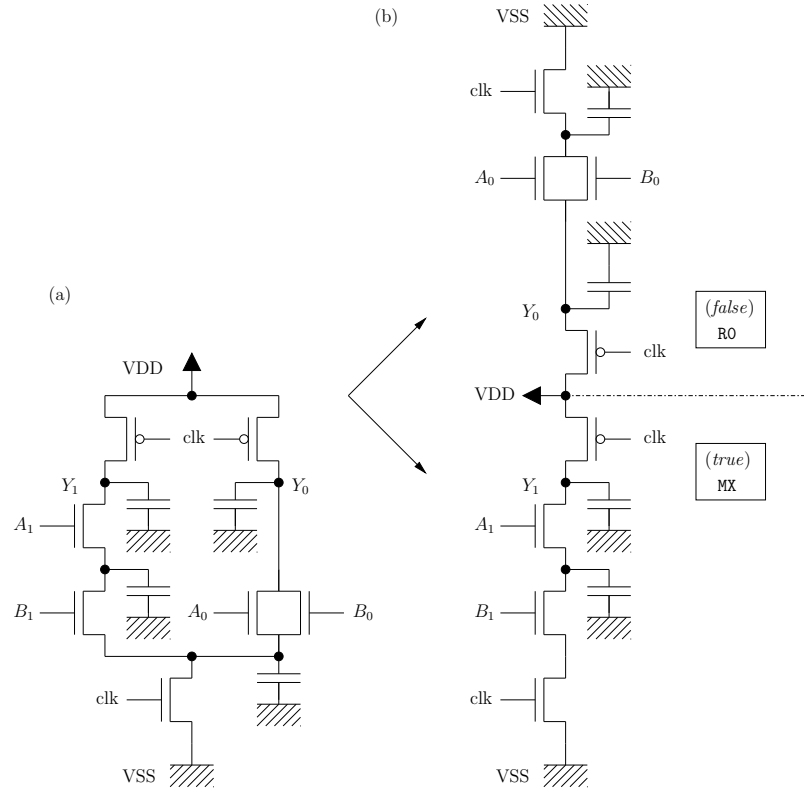


Figure 4.28: Transformation of a NAND gate implemented in SABL [103] (a) into two dual gates (b), for subsequent use in the “backend duplication” design flow.

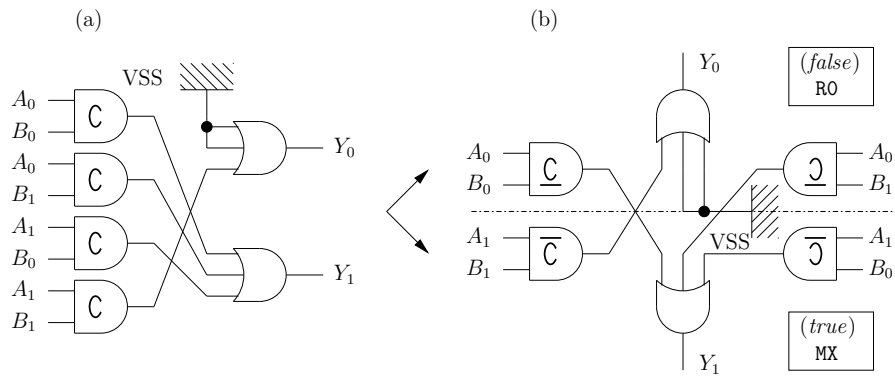
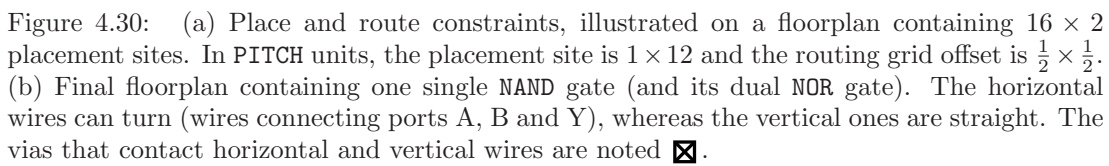


Figure 4.29: Transformation of a NAND gate implemented in SecLib [64] (a) into two dual gates (b). Notice that the two halves of the gate exchange signals.



Chapter 5

Conclusions

5.1 Summary of the dissertation

With the recent advent of the “information society”, most exchanges tend to become digital. As a consequence, information processing apparatuses become omnipresent. In this context of pervasive computing, security threats turn out to be major concerns. Until recently, the common attacks consisted in exploiting software bugs or network protocol weaknesses. Now that the computing devices (smartcards, tags, PDAs, PCs, *etc.*) get closer to the end-user, the implementation itself has become the target of attacks.

The goal of this dissertation is to investigate the strength of the attacks on the implementation of symmetrical block encryption algorithms, and to propose principled counter-measures against them. First of all, the attacks have been shown to be especially efficient on cryptographic implementations. We exhibited a paradox: some properties necessary to thwart *logical* attacks are shown to facilitate *physical* attacks. Then the two usual ways to realize the attacks, namely a correlation against the Hamming weight *versus* the Hamming distance, are compared. We conclude that the Hamming weight attack is not physically significative, even though it can lead to successful key extractions.

Some experiments have been conducted on an unprotected multi-mode DES crypto-processor. Attacks using the power consumption of the crypto-processor have been set up. The accurate analysis of the information leaks proved that the functional activity of gates could be exploited, as well as non-functional activity, commonly referred to as glitches. We even showed that attacks using the Hamming weight against substitution boxes could succeed because of a glitching activity occurring one clock period after the computation of the substitution boxes. Finally, we have simulated degenerated attack conditions: for instance, power traces were averaged over long period of time to mimic a low-frequency acquisition campaign. Even with this poor quality information, attacks were shown to succeed, albeit with a greater number of traces.

Knowing that attacks on the implementation were intrinsically strong, equally strong counter-measures have been studied. As it is impractical with the state-of-the-art technologies to remove the power dissipation, we have attempted to make power dissipation irrelevant to an attacker. The solution we study consists in designing circuits with appropriate symmetries to balance any otherwise unbalanced dissipation. The first step is to devise an elementary computing gate, which leads to the layout of a cell library called “SecLib”. Then we have shown that a secured backend flow, called “backend duplication” can be set up on top on existing legacy tools. A secured DES module has been designed to prove the feasibility of a geometrically symmetric circuit. The secured DES is usable in practice, because it embeds all the usual facilities: clock and reset signal bufferisation, test material based on scan chains and clock gating logic. Some properties, such as the signature of the substitution boxes input, have been confirmed thanks to the use of DES remarkable keys.

The secured DES is proved sound based on simulations. The next step is to evaluate the module *in silico*. If it resists attacks, then less expensive solutions could be investigated. Indeed, the use of “SecLib” with the “backend duplication” severely impacts the cost of the implementation, in terms of area, computation speed and power consumption. Otherwise, we would have proved that backend-level counter-measures against SCAs are irrelevant.

5.2 Perspectives

5.2.1 Open issues

Some open issues are still to be solved:

- Other SCAs (such as EMA) and FAs are still to be investigated on “SecLib”.
- Do inherently SCA-resistant algorithms exist?
- Will technological spreading of the circuit characteristics ruin all the geometrical counter-measures? This question is all the more important that situations exist where, if a single sample is defectuous, all the product line is corrupted.
- Will the evolution with Moore’s law foster attacks or counter-measures?
- Is there a potential for adiabatic logic or non-dissipating logics?

5.2.2 Going further

Some of the open issues identified in the previous sections have lead to the creation of national research projects. Some of them are listed below:

- OpenSmartCard <http://www.comelec.enst.fr/recherche/opensmartcard/>
- MARS <http://www.comelec.enst.fr/recherche/mars/>
- SAFE <http://www.comelec.enst.fr/recherche/safe/>
- SEMBAL <http://www.comelec.enst.fr/recherche/setin2006/>

Appendix A

Attackees/attackers technical details

A.1 The SecMat circuits family

Two ASICs have been developed to have a feedback from real circuits regarding attacks and counter-measures discussed in this manuscript. They are referred to as the “SecMat” (acronym for the french name “Sécurité du Matériel”) family. The version 1, represented in Fig. A.1(a), was used for the acquisition of power and electromagnetical traces, whereas the version $3/2$, represented in Fig. A.1(b), is dedicated to the comparison of SecLib with other logics.

A.1.1 SecMat frontend

A.1.1.1 SecMat V1 ASIC

The SecMat V1 experimental circuit is designed to validate countermeasures against the DPA (Differential Power Attacks.) It is made up of about two million transistors and has a silicon area of 3.10 mm^2 (4.00 mm^2 with the scribe lines and the alignment structures.) Its overall architecture is a bus-centric system-on-chip (SoC), described in Tab. A.1. Standardized modules, implementing the Advanced-VCi [108] interface, are plugged together onto a fixed priority bus mastered by an 8-bit 6502 CISC micro-processor. The processor boots a “monitor” from an embedded 2kb ROM and loads its program from the outside through an UART (up to 921 600 bauds) into a embedded 32kb RAM. The SoC is programmable in the C language. The main feature of the chip is the activation of the four cryptoprocessors — one AES and three DES — to lead DPA campaigns. It has been demonstrated interactively at the circuit exhibition collocated with the conference ESSCIRC’05.

The crypto-processors were willingly embedded within a SoC to avoid interferences between the encryptions and the pads activity: indeed, in the presented architecture, the cryptoprocessor’s program is loaded once, and then executes silently; the only pad that toggles is a trigger that is sent to the oscilloscope so that it properly synchronizes the acquisitions. This trigger is activated well before the encryption begins to ensure an optimal decoupling.

Measures obtained from an home-made smart-card based on a commercial micro-processor are given in Fig. A.2. The card is based on the ATMEga-128 processor from ATMEL; Its program is ISO-7816 compatible [2], so that acquisition campaigns can be launched via a standard smartcard reader. The smartcard is clocked thanks to the signal sent by the reader. Its frequency is 3.6864 MHz. Only a global power supply is available. The side-channel waveforms is thus highly distorted: oscillations at about 4 MHz deport some of the energy of one clock cycle onto the following ones. The consequence is that the power curves do not come back to

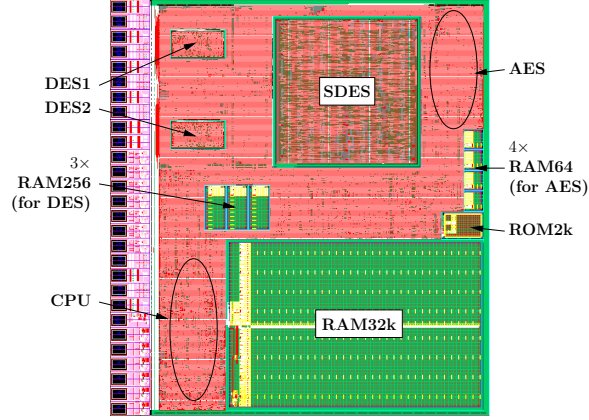
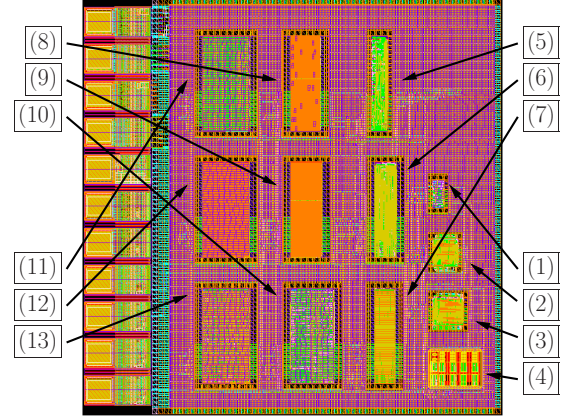
(a) *SecMat V1* (refer to [74, pp 62–63]).(b) *SecMat V3/2* (13 AES/SubBytes [72, 15]).

Figure A.1: Prototype ASICs developed in order to confront power models against actual measurements. (a) In SecMat V1, the target crypto-processor is labeled “DES2”. (b) In SecMat V3/2, all the SubBytes modules are targeted.

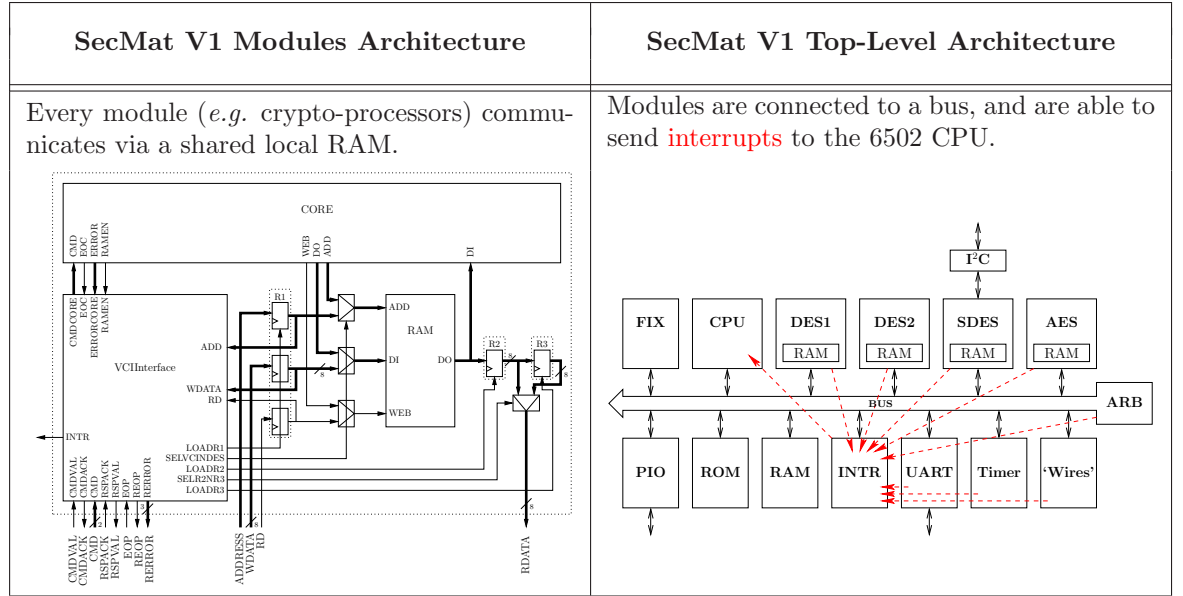


Table A.1: SecMat V1 System-on-Chip architecture.

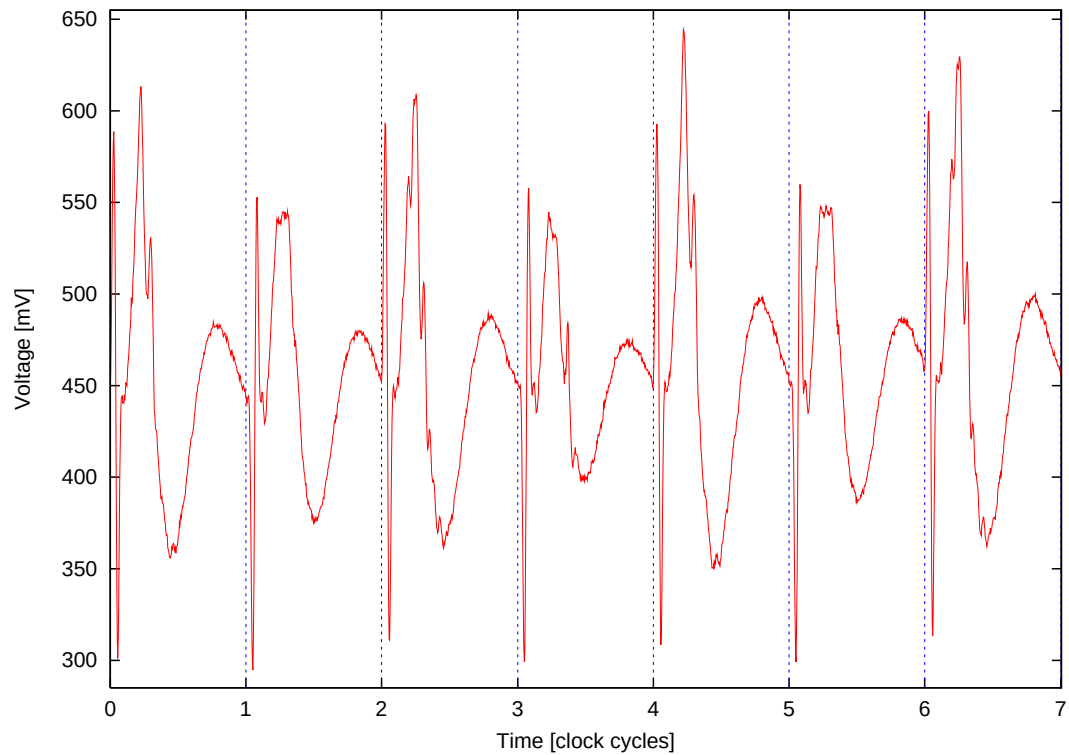
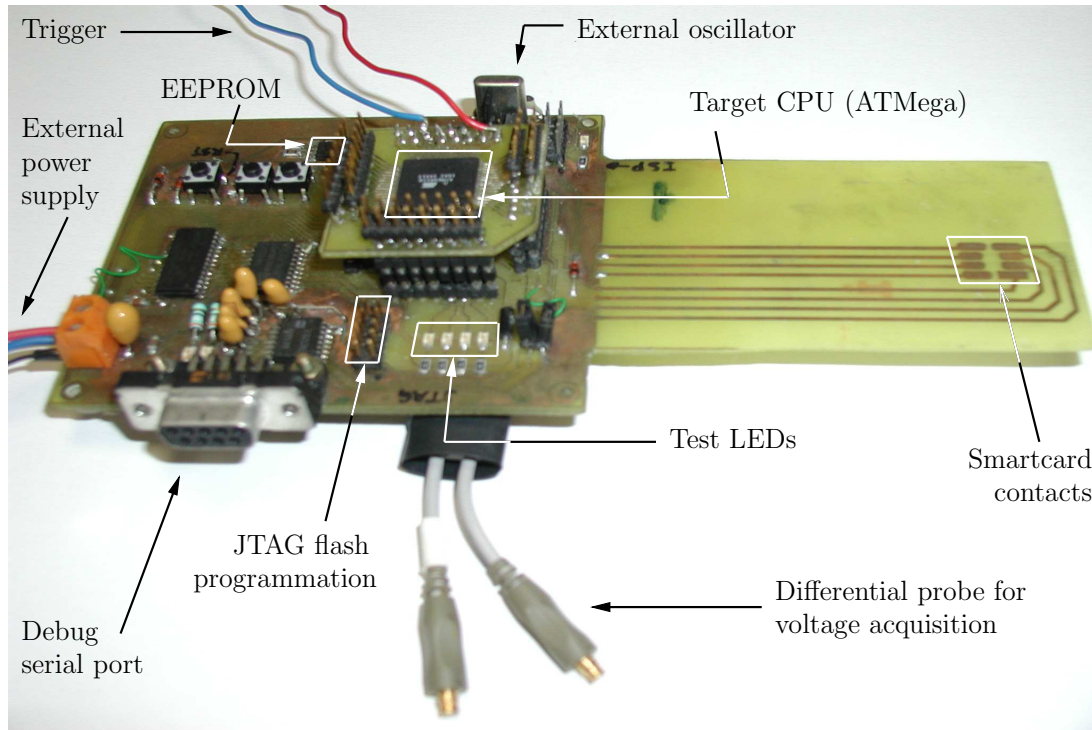


Figure A.2: (upper) Annotated picture of the experimental smartcard based on an ATmega processor. (lower) Seven clock cycles of the DES algorithm executed on the artisanal smartcard.

0 V after the rising edge of the clock. In spite of the poor quality of the setup, power analyzes are practical on this circuit. But as the implementation details are not totally known, accurate analyzes are not possible. This is one additional reason to use an ASIC design explicitly for the purpose of power analyzes.

In the SecMat V1 ASIC, the DES modules have been obtained from the same VHDL description (refer to Sec. 3.2.6), but are implemented into different micro-architectures. The most secure of those architectures uses a full-custom cell library, called SecLib (see Sec. 4.2), assembled into a synchronous dual-rail return-to-zero netlist (see Sec. 4.3.) This DES module is expected to feature a high level of immunity against side-channel attacks that exploit information leakage through the power consumption. The security of this module relies on a careful backend design that balance every possible dissymmetry [42, 102].

Two flaws in the DES layout may make its power analysis somehow easier than expected:

1. It does not have a clock tree. The clock net is thus heavily loaded (by $64 + 64 + 56 = 184$ DFFs), which implies that its variations are slow. Consequently, all the DES datapath registers probably consume more than with a sharp clock signal,
2. The control signals are not buffered, and thus lead to clearly visible signatures. Refer for instance to Fig. 3.24 at page 82 for a concrete illustration of so-called “**combinatorially-triggered**” dissipation.

Those two potential problems are readily solved for the next release of the SecMat system-on-chip.

A.1.1.2 SecMat V^{3/2}

The SecMat V^{3/2} circuit embeds thirteen versions of the “SubBytes” modules (substitution boxes used in the AES [72] encryption) in a 1.00 mm² silicon die. Its purpose is to enable a comparative evaluation of the several implementation of a same combinatorial block. The evaluation of sequential (synchronous or asynchronous) circuits is the task of the more complex SecMat (refer to [74, pp 62–63]) system-on-chip family.

Four libraries of cells were assessed:

1. Standard cell (130 nm low-leakage technology),
2. Read-Only Memory (“ROM”, generated by an automated generator),
3. WDDL [105], based upon the standard cell library,
4. SecLib [42], a custom secured QDI logic.

The two first libraries (standard cells and ROM) are unprotected, and can thus constitute references for the security evaluation. The four unprotected instances have different architectures, that are respectively:

1. Standard cell, look-up table [72, p. 16],
2. Standard cell, factored in GF(16²), as suggested by Vincent Rijmen [82],
3. Standard cell, decode/permute/encode, as described by Guido Bertoni [15],
4. Generated low-power contact-programmable ROM.

The SecMat V^{3/2} circuit is still in foundry and has not been characterized yet.

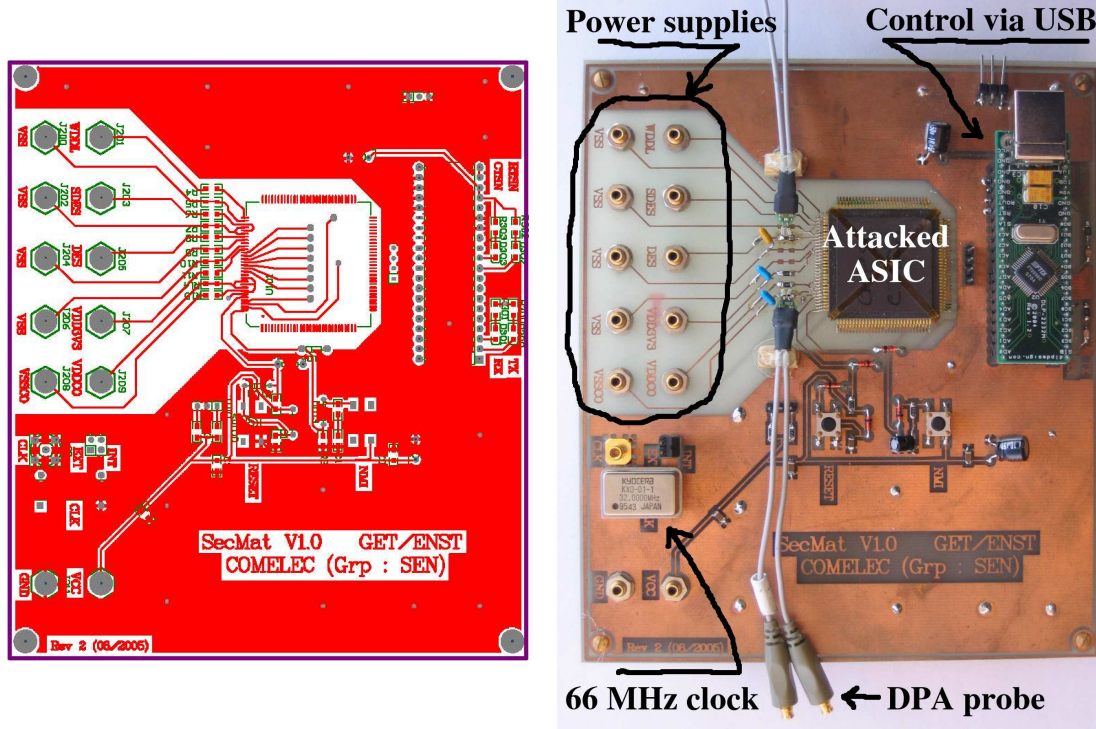


Figure A.3: PCB plan (*left*) and photograph (*right*) of the attack board front view, with the SecMat V1 ASIC in exergue.

A.1.2 SecMat backend

The SecMat circuits were synthesized with Cadence `pks_shell` and placed-and-routed with Cadence `encounter`. The modules to cryptanalyze (DES in SecMat V1 and SubBytes in SecMat V^{3/2}) were powered by a dedicated supply pair, that convey the $v_{ss}=0$ volt and $v_{dd}=1.2$ volt voltages directly into to the co-processor, equipped with its own power ring. The private voltage of every co-processor is noted V , whereas the circuit's core voltage is noted U . In normal operating conditions, $V=U=1.2$ volt. For the sake of attacks, the voltages may be tuned, as shown in Tab. A.2.

The process is a low-leakage (hence high threshold voltage) 130 nm technology with six metal layers (M1–M6). The chips are fabricated through the multi-wafer projects of the CMP, run S12C5_1 (for V1) and run S12C6_3 (for V^{3/2}).

A.2 The attack boards

The SecMat V1 circuit is placed on a motherboard that is controllable remotely via a single USB socket. The serial $\leftrightarrow$ USB conversion is delegated to an FTDI DLP-2232M module. The SEC MAT circuit monitor is functional and can execute arbitrary code injected from a PC. The attack motherboard is shown in Fig. A.3.

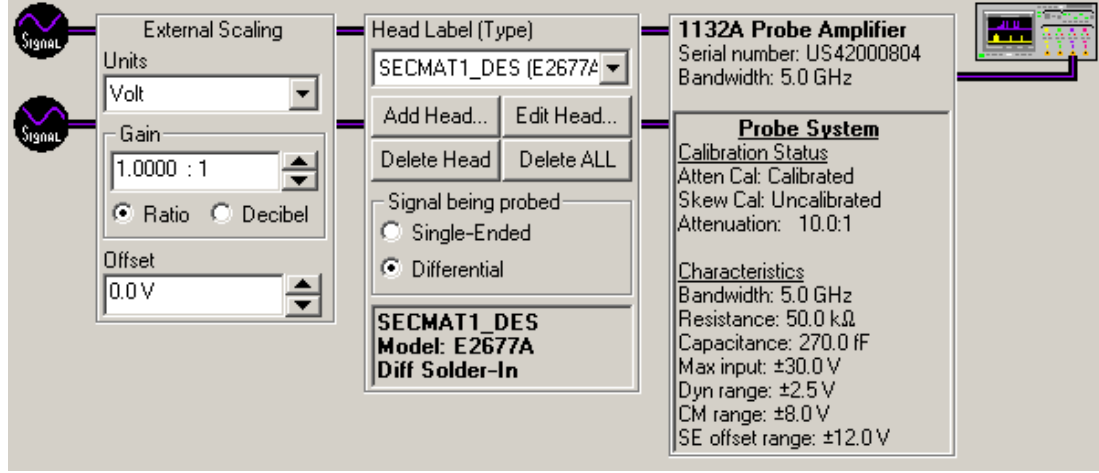


Figure A.4: Differential voltage acquisition probe configuration.

A.3 The acquisition setup

The acquisition apparatus is an INFINIUM 54855A oscilloscope from [Agilent](#). The probes' model is 1132A, featuring a bandwidth of 5 GHz. The E2669A differential connectivity kit was used. The power traces shown in this document were acquired with a solder-in connector. The configuration of the acquisition probe is depicted in Fig. A.4. The overall experimental setup is shown in Fig. A.5. It is suitable for:

- **power attacks**, thanks to the differential voltage probe, and
- **EMA**, thanks to an *ad hoc* antenna, preceded by a 32 dB active amplifier.

A.3.1 Optimal power traces acquisition experimental conditions

The success of a DPA depends both on a proper acquisition setup and on an advanced traces processing (confidential information extraction.) This short notice studies the optimal acquisition setup based on an extra resistance added by an attacker to access the power side-channel. The traces shown in this manuscript were acquired with a resistance $R = 11 \Omega$, but the results reported here-after show that a wide range of values are actually suitable.

Two wafer doping styles exist: *double* and *triple-well*. In those two cases, the ground is common to all the transistors. On the contrary, the highest voltage is provided only to the specific rows containing P-MOS transistors in an N-well. As the polarisation of P-MOS transistor can be decoupled, the spy resistance is placed on the VDD-side of the chip.

A.3.1.1 Discussion about the pros and the cons of some alternatives

Side-channels attacks exploiting the power dissipation of a working cryptoprocessor need an access to the power. A common experimental setup requires to divert some of the power consumed by a chip into a “spy” resistance. The choice of the resistor value R is up to the attacker.

If it is chosen small, the amplitude of the traces collected is small, which leads to a poor-quality attack. Indeed, the state-of-the-art oscilloscopes can only use the full 8-bit vertical precision if calibers 10 mV or above are used (signal dynamics ≥ 80 mV.)

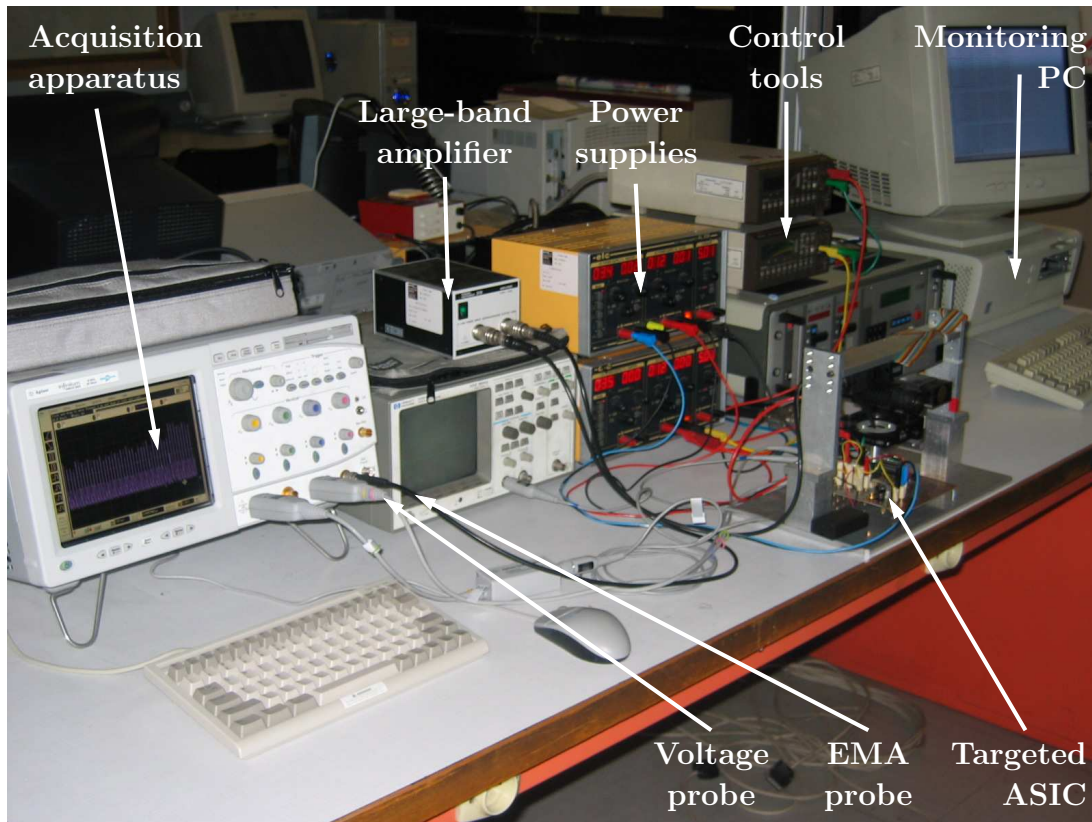


Figure A.5: Experimental setup for SCA models validation against practical measurements.

At the opposite, if R is chosen too large, the circuit is disturbed by the voltage drop across the resistance. In a realistic scenario, an embedded analog power controller would check for the stability of the voltage delivered to the circuit. In this study, we assume that such protection does not exist. It is thus possible to choose an arbitrary resistor R .

The maximum value R_{max} is that where the circuit begins to malfunction. Indeed, at that point, the power traces do not correspond to the expected computation, hence any correlation attack is doomed to fail. It is however a way to inject errors during the processing, which enables fault attacks.

If the resistor is kept beyond R_{max} , the circuit remains functional. An attacker can thus decide arbitrarily for $R \in]0, R_{max}[$. Now, for R close to R_{max} , the electronic gates are under-supplied, and thus consume less power (that is proportional to the square of the voltage). As a consequence, an higher signal is measured for a weaker phenomenon. Thus, there exists an optimal value R_{opt} for R .

The value R_{opt} could be defined as the value for which the measured signals have the largest amplitude. However, the real challenge is to increase the strength of the DPA. Thus the actual criterion is the number of traces needed to reach a given signal-to-noise ratio for the DPA.

A.3.1.2 Power measurements

Two experiments were carried out on the DES crypto-processor datapath (that has a separate power supply V) embedded into the SecMatV1 ASIC (powered by U .) The datapath of the DES module is described in Sec. 3.2 and in [101].

In the first experiment, the voltage is dropped from the nominal value, 1,2 V, until the results of the DES cryptoprocessor become false. In the second experiment, the DES datapath is supplied with its nominal 1,2 V, but a variable resistor R is inserted between the power supply and the power of the DES datapath. The results are given in the table A.2 below.

It is noticeable that when DES fails close to the maximum resistor R_{max} or the minimum voltage V_{min} , the result is not full-zero. The errors actually concern the loading of the message into the RAM. The faulted ciphertext is actually almost correct, apart from sparse errors. The errors always lead to lower values, for the following partial order: $a < b \Leftrightarrow \forall i, (a \& 2^i) < (b \& 2^i)$. This fact is shown in Tab. A.3. The errors thus probably arise from the fact that 0.6 V cannot allow the RAM, powered at about twice this voltage, to distinguish between the actual digital values “0” or “1”. Actually, for the DES processor to remain functional, two constraints must be met simultaneously:

$$\frac{1}{2} \times U < V < 2 \times U.$$

The first strict inequality makes sure that the core properly interprets the “1”s from the DES module and *vice-versa* for the second. There is no risk that “0”s be misunderstood, because the DES module and the core share the same ground; this equipotential is the common PWELL made up of the silicon substrate (*aka the wafer’s bulk.*)

The faults induced by under-voltage and/or over-voltage are referred to as a “semi-invasive” attack¹.

Figure A.6 shows the average power consumption of the DES datapath when V varies, while U is fixed at 1.2 volt. The power dissipation has been measured irrespectively of the fact that the DES datapath works properly or not. Figure A.6(a) shows that the dissipation is almost null below the $V \sim 0.3$ volt. The reason is that there can be no activity if the transistors remain blocked; this happens when the power supply is below the absolute value of the threshold of all the transistors: $V < \min(V_{ThN}, -V_{ThP})$ [50, chap. 5]. In the so-called ‘typical’ characterization corner, the thresholds are equal to: $V_{ThN} = 295$ mV and $V_{ThP} = 367$ mV. Now, in Fig. A.6(b), the dissipation increases significantly when V is over 1.5 volt, because the P-MOS transistors

¹See the taxonomy of Sergei P. Skorobogatov at http://www.cl.cam.ac.uk/~sps32/semi-inv_def.html.

Table A.2: Measures of the correctness probability of DES encryptions for $\text{DES}(m=\text{"0011223344556677"}, k=\text{"0011223344556677"}) = \text{"e4ff10812363072d"}$ (without faults.)

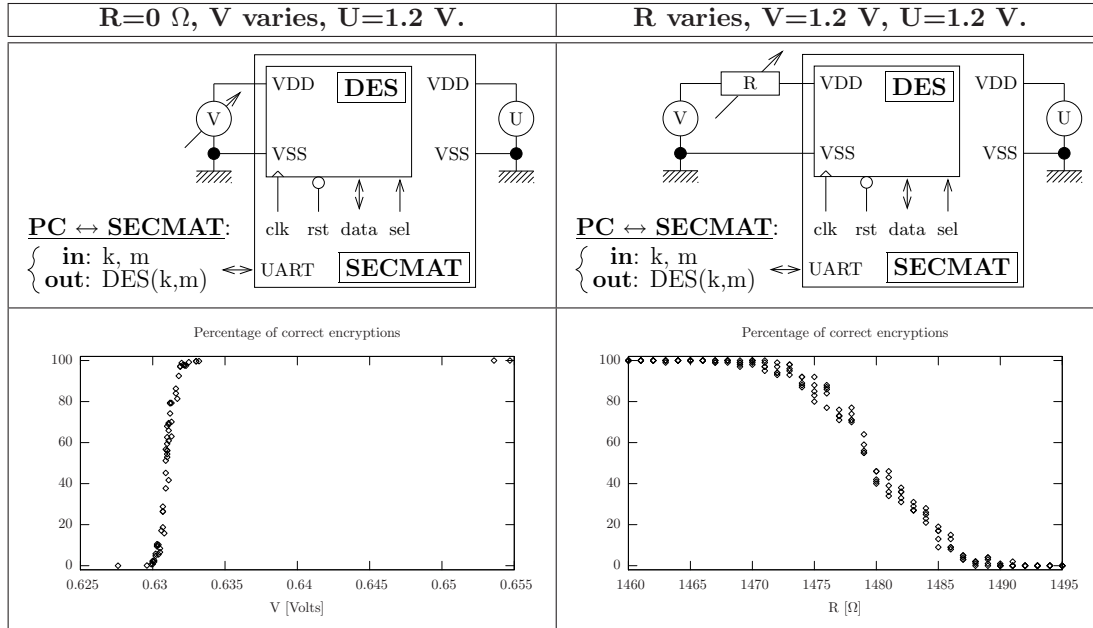


Table A.3: Errors occurring for every nibble of faulted ciphertexts based on 1 000 000 encryptions (correct results percentage is 35 %).

Correct nibble	Faulted nibble ($\prec$ Correct one)
e	2, a
4	none
f	0, 1, 3, b
f	4, 6, e
1	0
0	none
8	0
1	0
2	0
3	2
6	0, 2
3	2
0	none
7	6
2	0
d	4, c

of the input gates become permanently on: $V_{GS} > V_{ThP}$. The figure A.7 illustrates the short-circuits that can be caused in this situation. This effect is important because it can trigger thermal runaway in the circuit causing its destruction. In addition, there is a risk for the gates driven by the DES datapath to start a latch-up effect in the “low-voltage” part of the circuit. This latch-up would propagate to the whole chip (CPU, memories, *etc.*), which would probably destroy it. For these reasons, V has not been set too much over 2.2 volt [11].

A.3.2 The acquisition software

The architecture of the **acquisition** software is depicted in Fig. A.8. The **main** function (entry point in C) builds an **acquisition** object, that undertakes the following two actions:

1. build a **cmdline** object, and customize it with the user options (call to **parseMain**),
2. build an **attack** object, that is able to **launch** the acquisition, that has collaborate an **attacker** and an **attacker**. An attacker is for instance a single (**gpib_device_54850_3**) or a multi-modal (**gpib_device_54850_3_4**) oscilloscope, whereas an attacker is a device embedding dedicated crypto-processors (**attacker_crypto**) or running cryptographic software (**attacker_rta**, attacked via an **rta**, *i.e.* a register transfer attack.)

Then, the typical attack loops are represented in the sequence diagram for the acquisition shown in Fig. A.9. It is worth noticing that all the events are asynchronous, in the sense that the acquisition does not fail if random delays are inserted between them. The only strict timing constraint to be respected is the constant time between the trigger edge and the “consuming event” (encryption.)

As illustrated in Fig. A.9, one central PC controls the whole acquisition process. It can be programed to launch an acquisition remotely, for instance from the internet via a secure shell (**ssh**) connection.

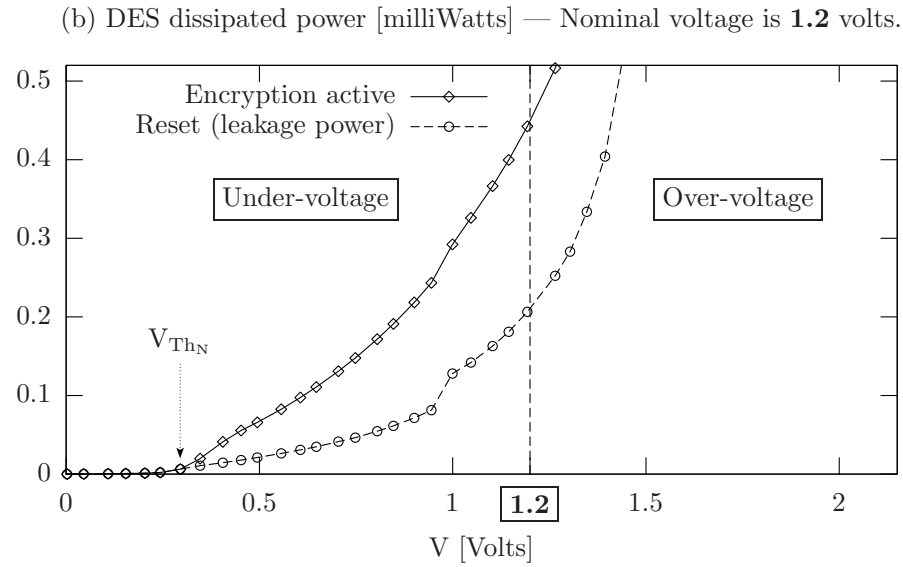
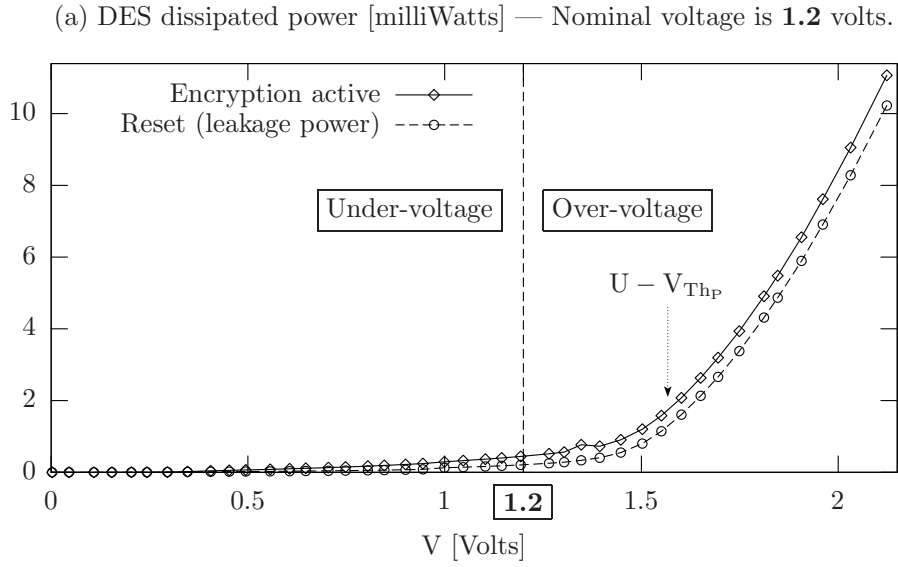


Figure A.6: (a) Power dissipated by the DES datapath, when U is fixed at 1.2 volt. (b) Zoom on the same graph, near the nominal voltage.

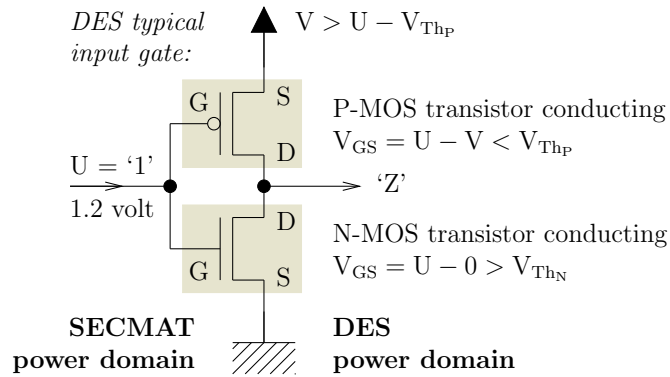


Figure A.7: Short-circuit occurring when the logical input '1' is applied from the SecMat core to the DES module, powered with $V > U - V_{ThP}$. The 'Z' states inside the DES module might propagate metastability.

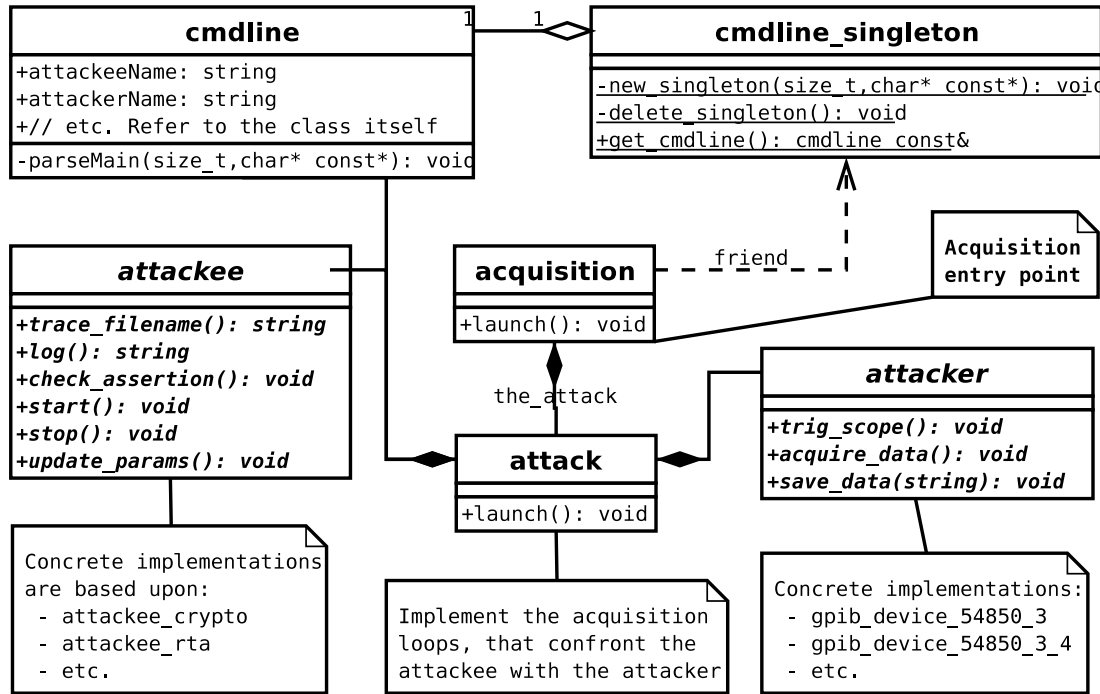


Figure A.8: UML class diagram for the acquisition application (also refer to the concept of Fig. A.9.)

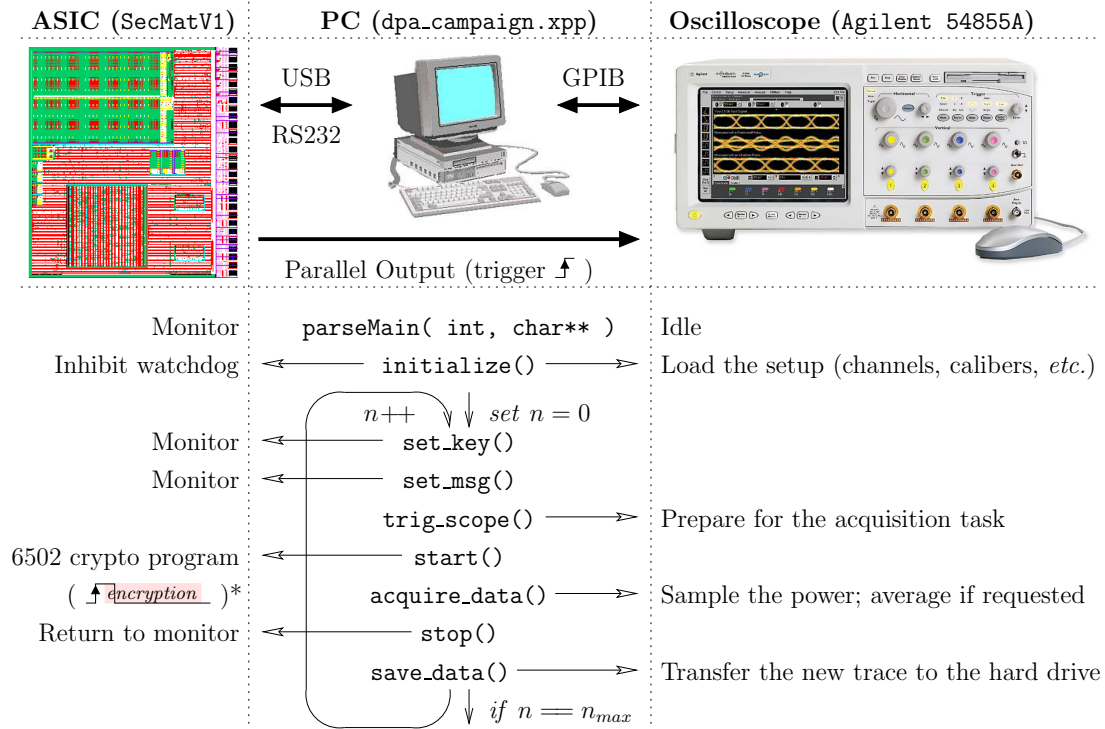


Figure A.9: Typical acquisition sequence diagram, for an attack (conducted by a PC running the `acquisition_campaign.xpp` software) that involves both an attackee (the ASIC SecMatV1 of Fig. A.1(a)) and an attacker (Agilent model Infiniium 54855A.)

Appendix B

Power Traces on the DES Co-Processor of SecMat V1

The noise induced by the acquisition apparatus and the variation of the environment conditions is evaluated for several built-in averaging of the oscilloscope. The same message is encrypted 1 000 times with the same key. The 1 000 traces are made up of 20 000 data points, acquired at the rate of 20×10^9 samples/s, corresponding to 32 clock periods of a circuit running at 32 MHz. The vertical caliber is set to 50×8 mV. The average standard deviations are given in Tab. B.1 for averaging ranging from 1 (*i.e.* no averaging) to 4096 times. The duration of the acquisition of a trace is about proportional to the number of averages. It must be noted that 44 % of the time is spent by post-processing between every trace acquisition; put differently, the oscilloscope is triggered only 66 % of the acquisition duration.

It appears that the experimental standard deviation decreases by a factor $1/\sqrt{2}$ when the averaging is doubled, in the range $[0, 256]$. This observation is in line with the official resolution documentation from Agilent. Without averaging, the traces resolution is 8 bits. Using the built-in averaging capability of the oscilloscope, the resolution can reach 12 bits, as shown in Tab. B.2. Beyond the amount of 256 averaging, a residual error cannot be corrected. The reasons may be:

- the quantification noise of the oscilloscope analog-to-digital converters (ADCs) or
- the variation of the experimental conditions, since the acquisitions take many hours when large averages are requested.

It is worth noticing that the averaging rate is not limited by the frequency of the trigger

Table B.1: Acquisition characteristics for various averaging factors.

Averaging	2^0	2^1	2^2	2^3	2^4	2^5	2^6
Standard deviation [mV]	6.41	4.47	3.18	2.22	1.59	1.15	0.82
Acquisition time [s/trace]	0.317	0.334	0.456	0.628	0.960	1.656	2.966

Averaging	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
Standard deviation [mV]	0.59	0.44	0.37	0.39	0.87	0.57
Acquisition time [s/trace]	5.624	10.884	21.469	42.596	84.785	169.268

Table B.2: Agilent’s 54855A acquisition resolution, quoted from the oscilloscope datasheet.

Averaging	2 ⁰	2 ²	2 ⁴	2 ⁶	2 ⁸
Resolution [bit]	8	9	10	11	12

signal, but by the oscilloscope. The trigger is generated by software, and it takes about 853 CPU cycles to loop; so, for instance, when SecMat runs at 32 MHz, the period of the trigger is 26.66 μ s. This period could be optimized, but this is useless, because at best (*i.e.* for 4096 averages), only one trigger produced by SecMat out of about 1000 actually triggers the oscilloscope.

The limitation may be due to:

- the slow arming rate of the oscilloscope or
- the computation-intensive averaging algorithm itself.

The algorithm used by the oscilloscope to achieve an averaging by N consists in:

- initializing an average wave “ $\bar{T}$ ” to zero: $\bar{T} \leftarrow 0 \cdots 0$,
- accumulating the new trace T , with the following weighting: $\bar{T} \leftarrow \frac{n-1}{n}\bar{T} + \frac{1}{n}T$, for $n \in [1, N]$,
- then, if further samples are available, the average is updated according to: $\bar{T} \leftarrow \frac{N-1}{N}\bar{T} + \frac{1}{N}T$.

The dependence of the average on the traces is thus infinite, at least in theory. In practice, the series $\left\{ \left(\frac{N-1}{N} \right)^i \right\}_{i \in \mathbb{N}}$ quickly decreases below the minimum floating point precision.

A fair trade-off between acquisition speed and vertical resolution is to use a 64-times average. All the traces shown in this manuscript have been averaged 64 times, with exception of Fig. 3.21, acquired with the maximal 4096-times averaging.

The power traces are reproducible as long as the board is not modified. Otherwise, the signal’s shape is altered.

Acquisitions are said to be reproducible if the mean standard deviation of averaged traces acquired on different days is similar to the standard deviation of the acquisitions realized on one day. Practically speaking, acquisition campaigns of 1000 traces averaged 64-times were realized on four different days. The four mean traces $\bar{T}_i, i \in \{1, 2, 3, 4\}$ were computed, and the mean standard deviation is equal to: 0.96 mV. As already reported in Tab. B.1, the standard deviation of every individual campaign is 0.82 mV. More precisely, the four deviations are respectively 0.82, 0.81, 0.82 and 0.82 mV. As 0.96 mV $\approx$ 0.82 mV, the campaigns are reproducible.

B.1 Hamming weight *vs* Hamming distance differential traces

This section reports an acquisition campaign realized on the DES hardware encryption of the ASIC SecMat V1. The architecture of the crypto-processor is extensively described in chapter 3. The campaign consists in the acquisition of 81089 traces with a constant key, `jexjexje` in

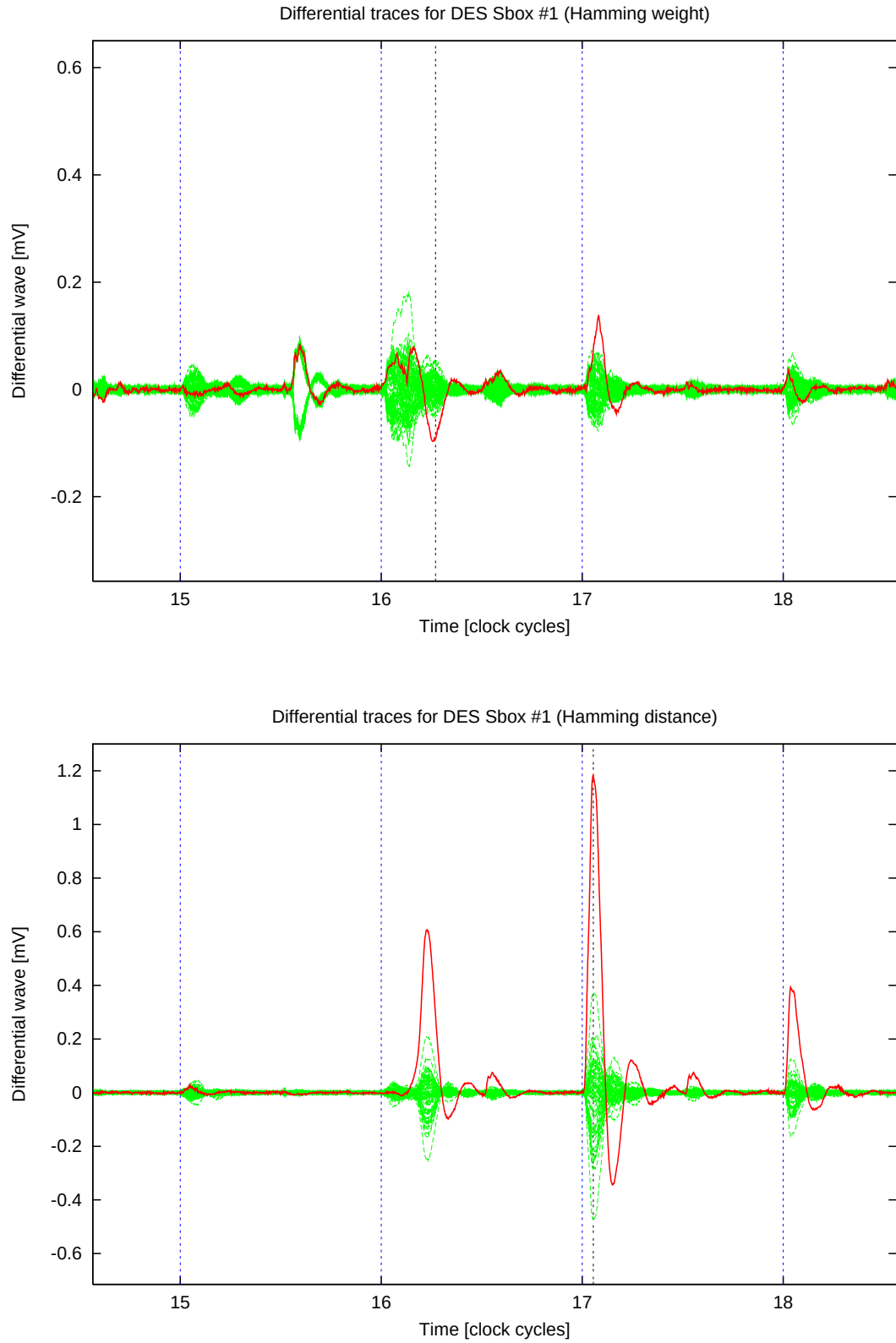
ASCII or 0x6a65786a65786a65 in hexadecimal. The spying resistor is on the VDD side of the power supply, its resistance is 11 Ω , and the voltage is the nominal value of 1.2 volts.

The eight figures B.1 to B.8 are the differential traces obtained for the $2^6 = 64$ hypotheses on the key on each sbox. The figure B.1 was already printed as Fig. 2.21 at page 46, but is printed in this appendix once again for the collection of the eight sboxes to be complete.

B.2 DPA signal-to-noise ratios on DES

This section is intended to present the DPA signal-to-noise obtained with the traces collection described in former Sec. B.1. The plot is **red** when the key guess is correct, and **green** otherwise.

Notice that the weird variations in the SNR of the sbox #8 can be explained by the choice of the date at which the SNR is evaluated. The SNR computation algorithms looks for the date that optimizes the SNR. It actually happens that this date is oscillating between two values, as reported in Fig. B.17. A smoother SNR curve is obtained with an *a priori* choice for the date.

Figure B.1: Hamming weight for sbox #1 (*upper*) — Hamming distance for sbox #1 (*lower*).

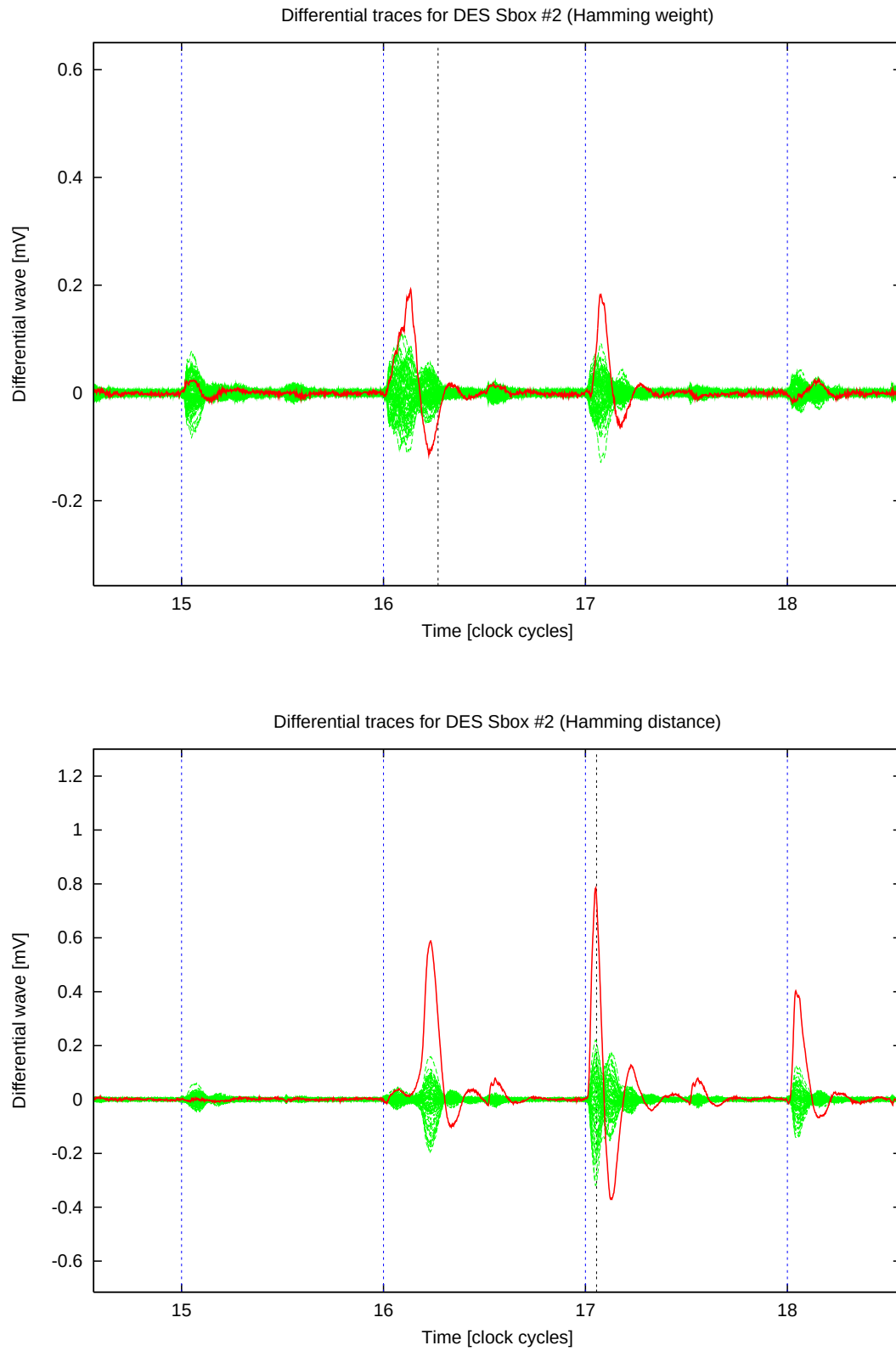
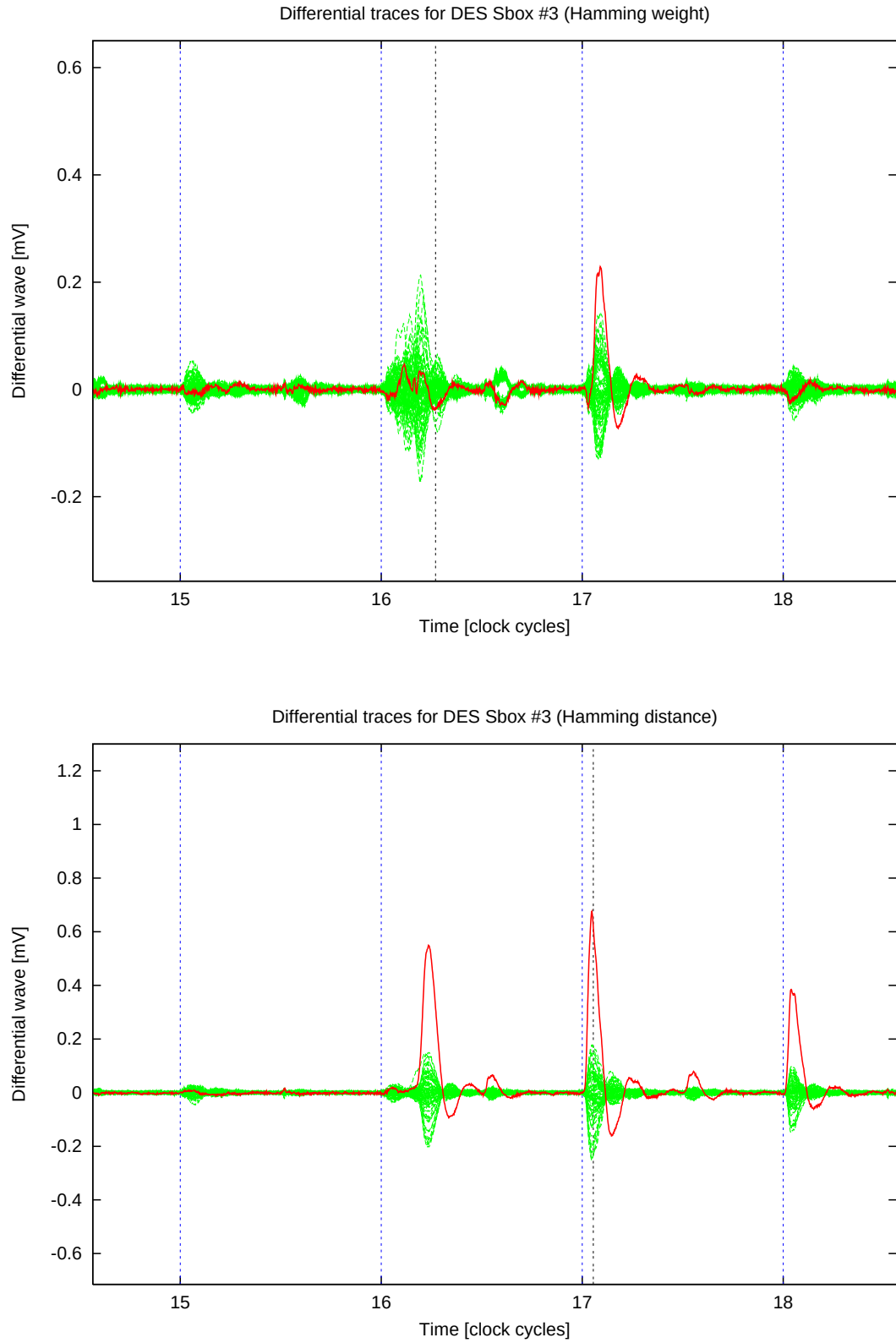


Figure B.2: Hamming weight for sbox #2 (*upper*) — Hamming distance for sbox #2 (*lower*).

Figure B.3: Hamming weight for sbox #3 (*upper*) — Hamming distance for sbox #3 (*lower*).

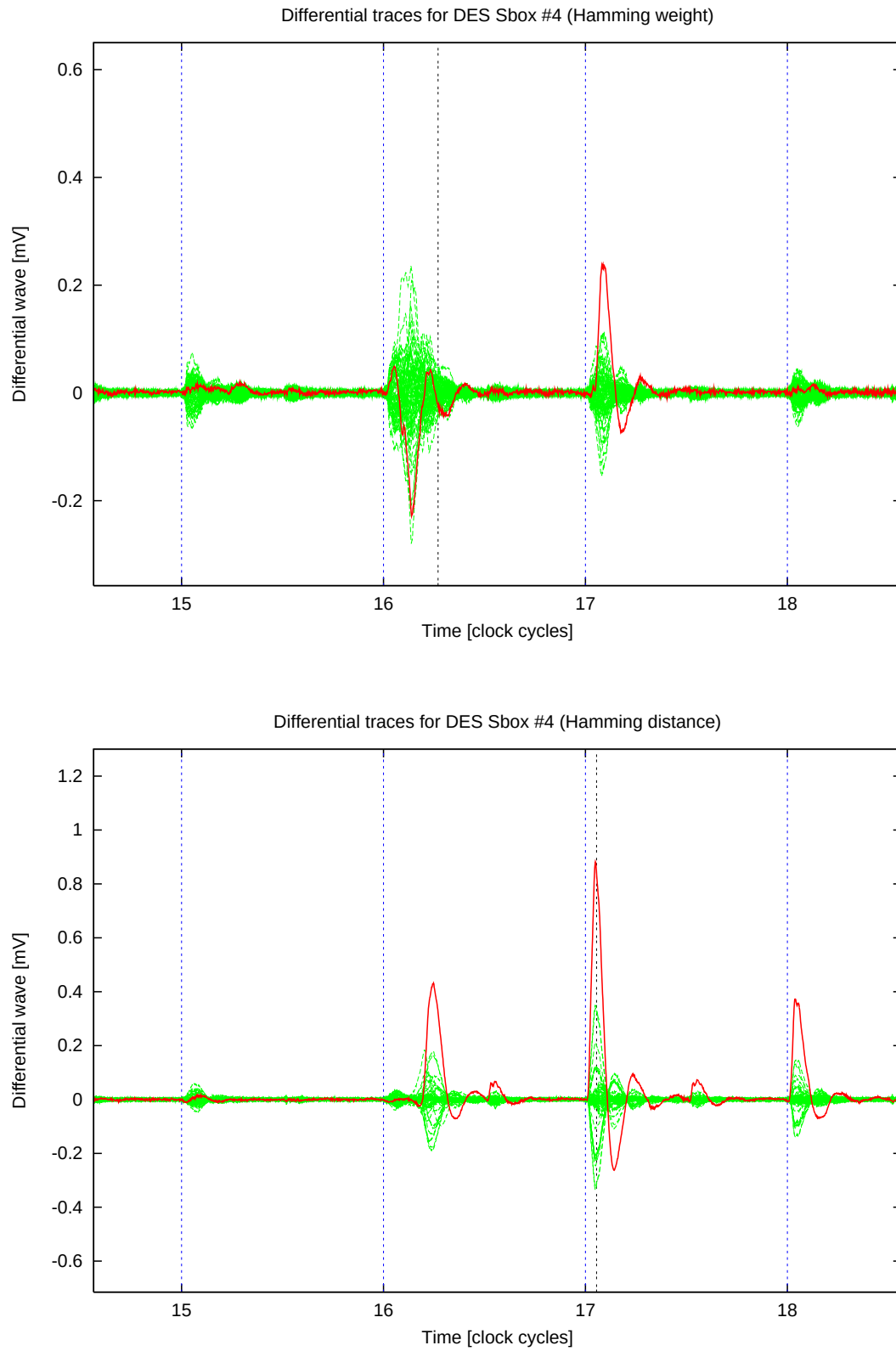
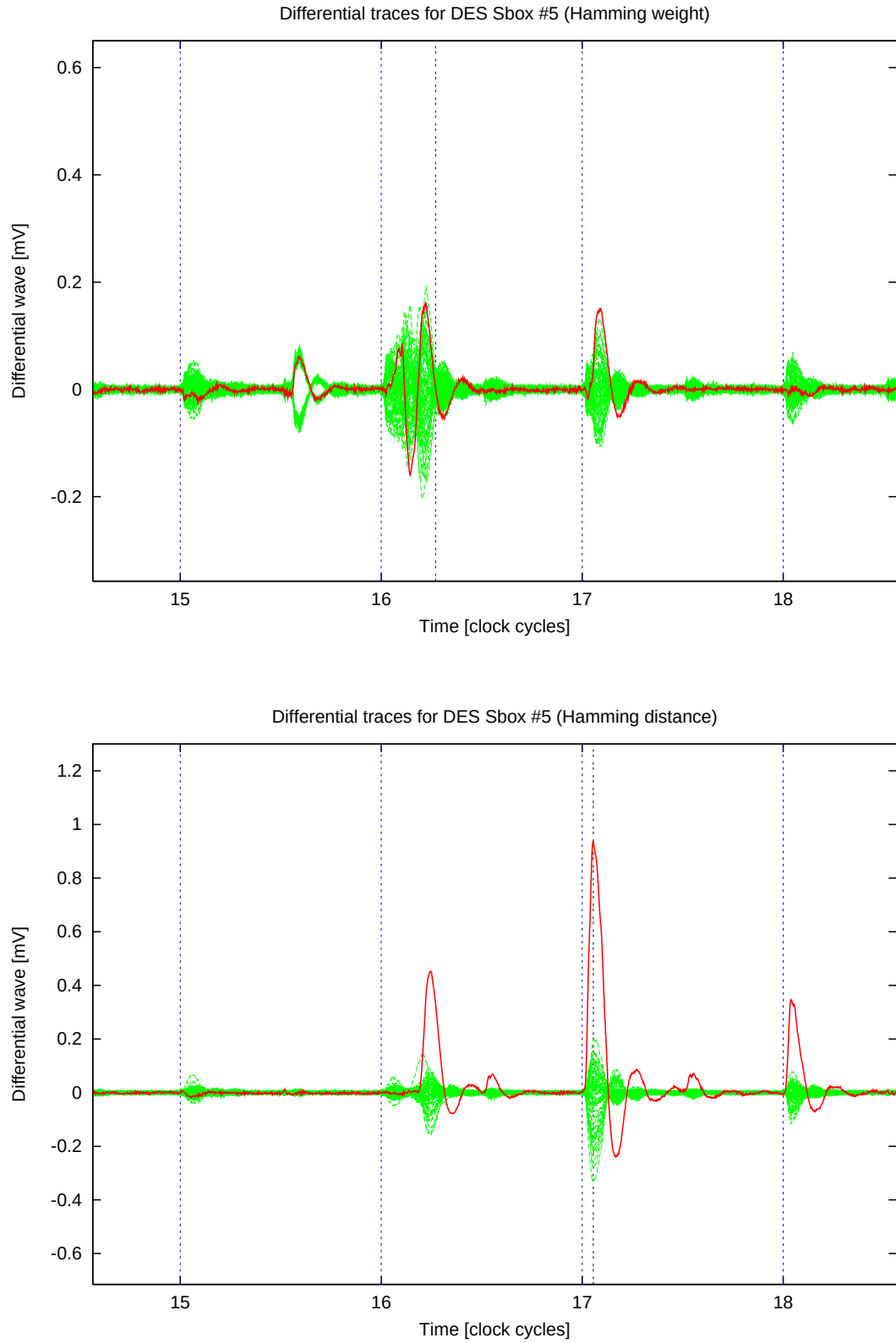


Figure B.4: Hamming weight for sbox #4 (*upper*) — Hamming distance for sbox #4 (*lower*).

Figure B.5: Hamming weight for sbox #5 (*upper*) — Hamming distance for sbox #5 (*lower*).

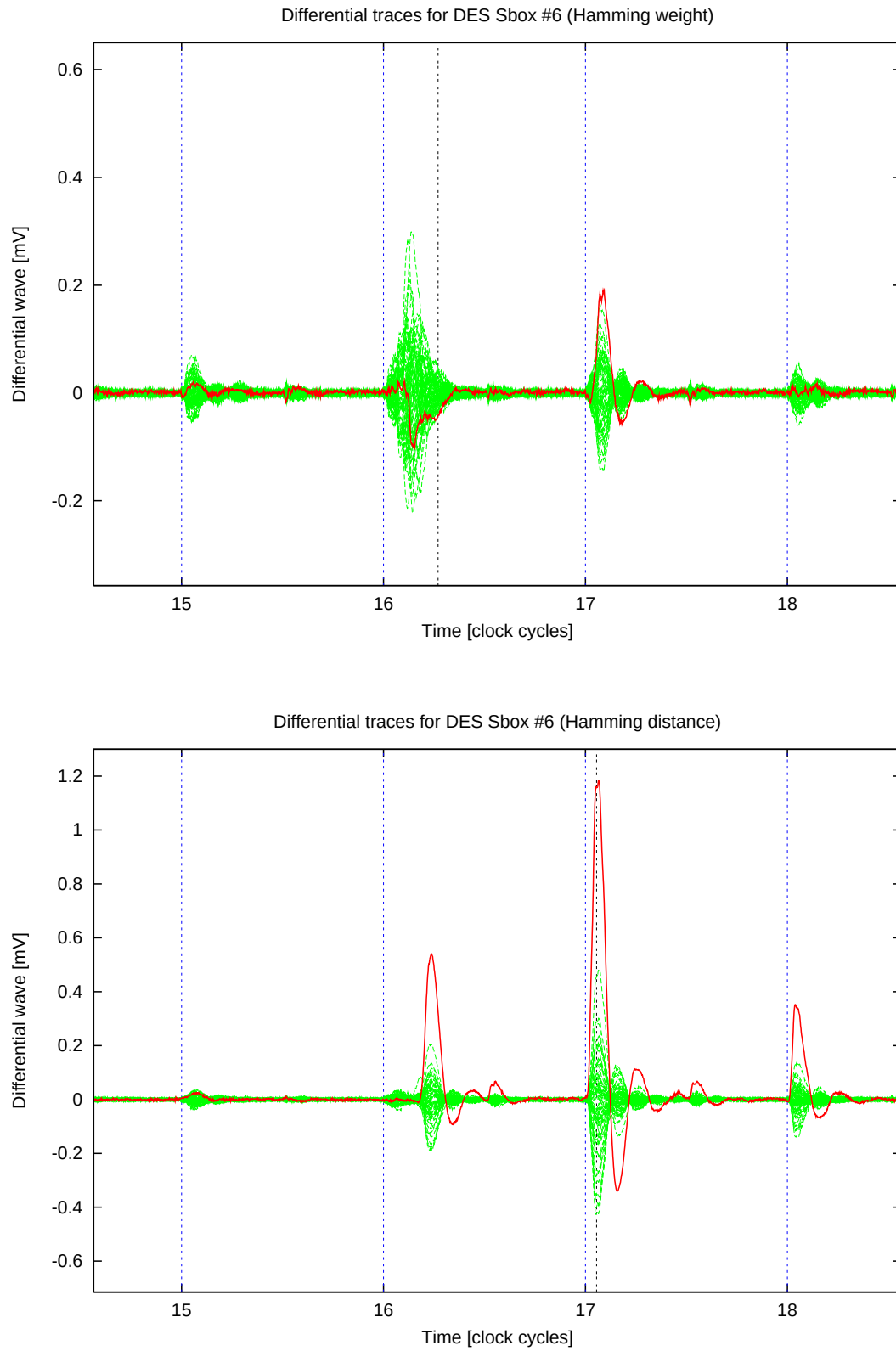
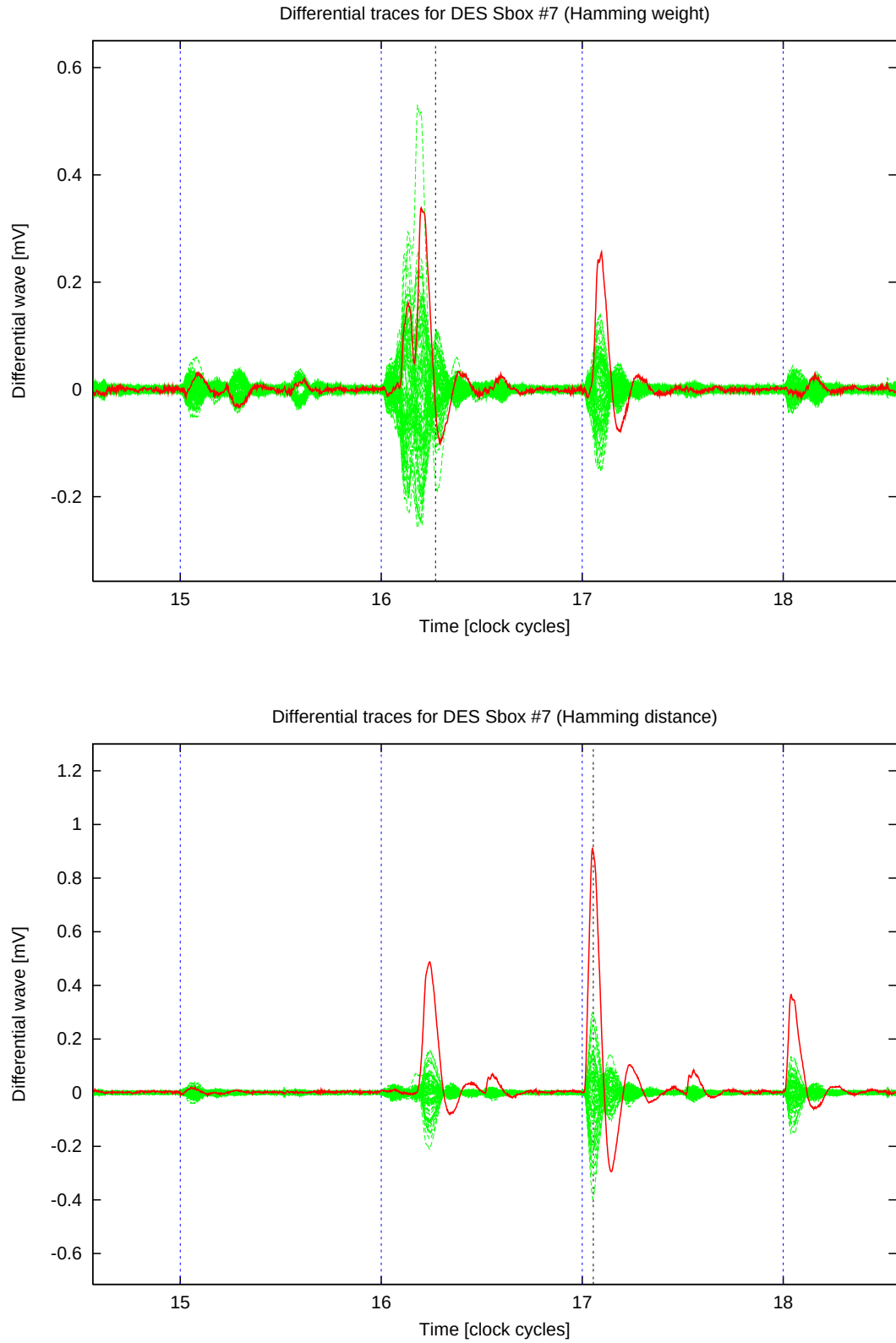


Figure B.6: Hamming weight for sbox #6 (*upper*) — Hamming distance for sbox #6 (*lower*).

Figure B.7: Hamming weight for sbox #7 (*upper*) — Hamming distance for sbox #7 (*lower*).

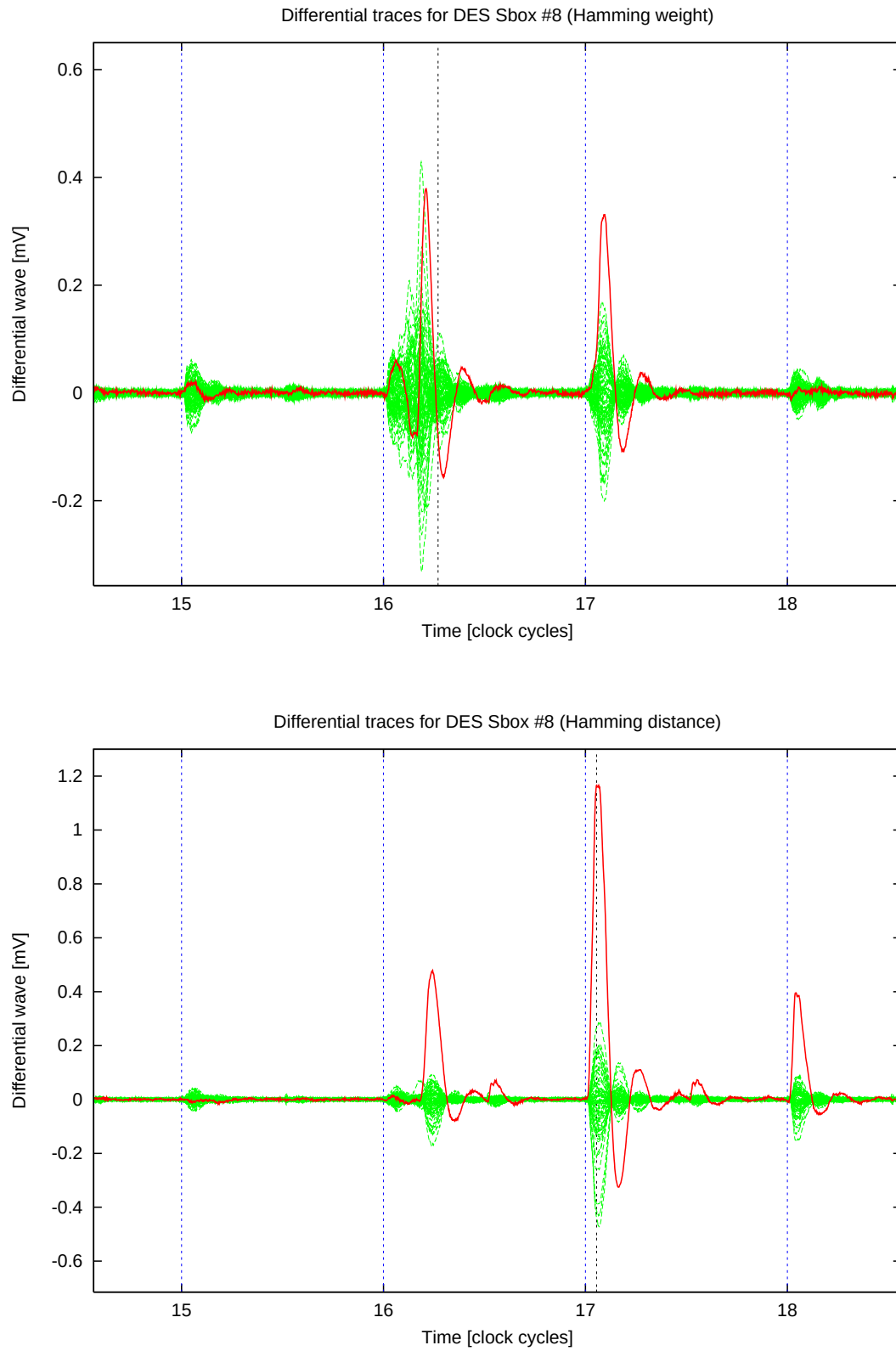


Figure B.8: Hamming weight for sbox #8 (*upper*) — Hamming distance for sbox #8 (*lower*).

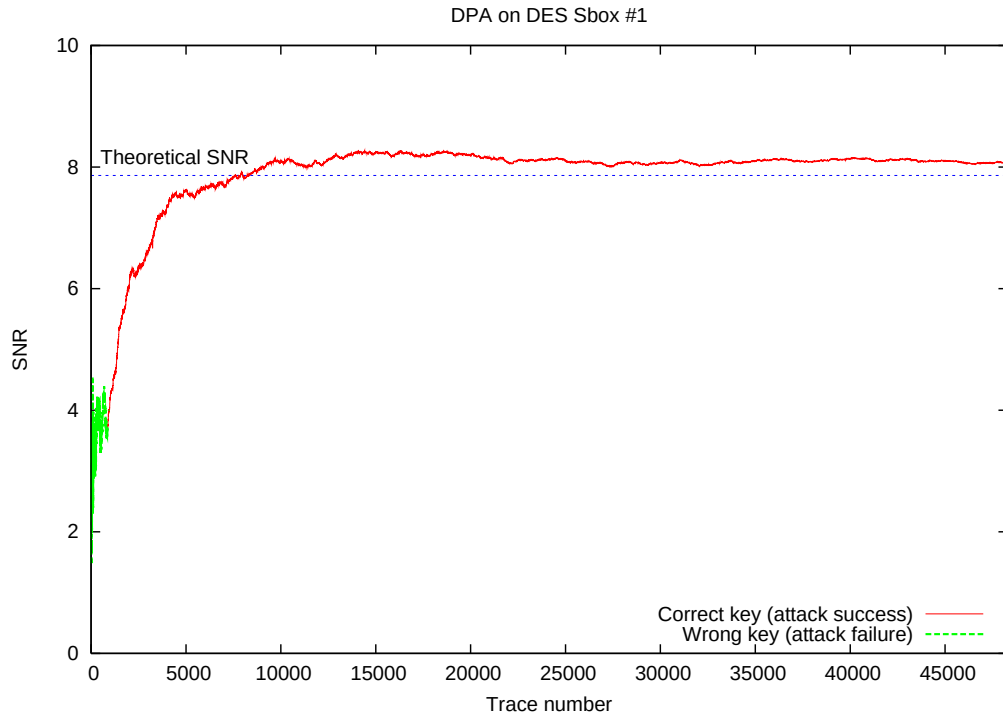


Figure B.9: SNR of the DPA on DES sbox #1.

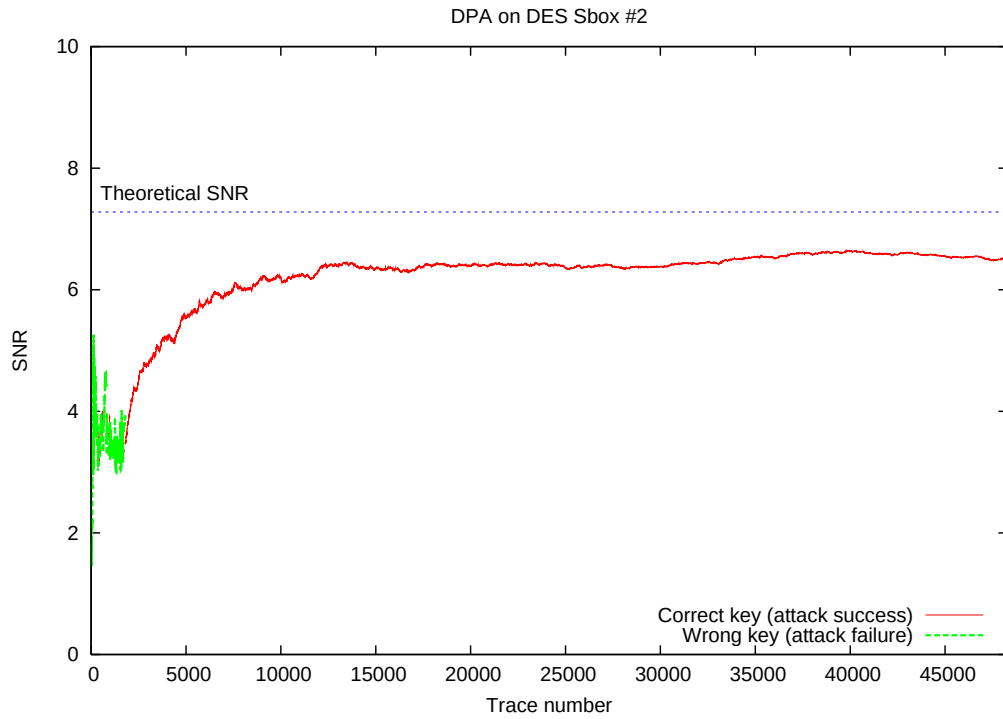


Figure B.10: SNR of the DPA on DES sbox #2.

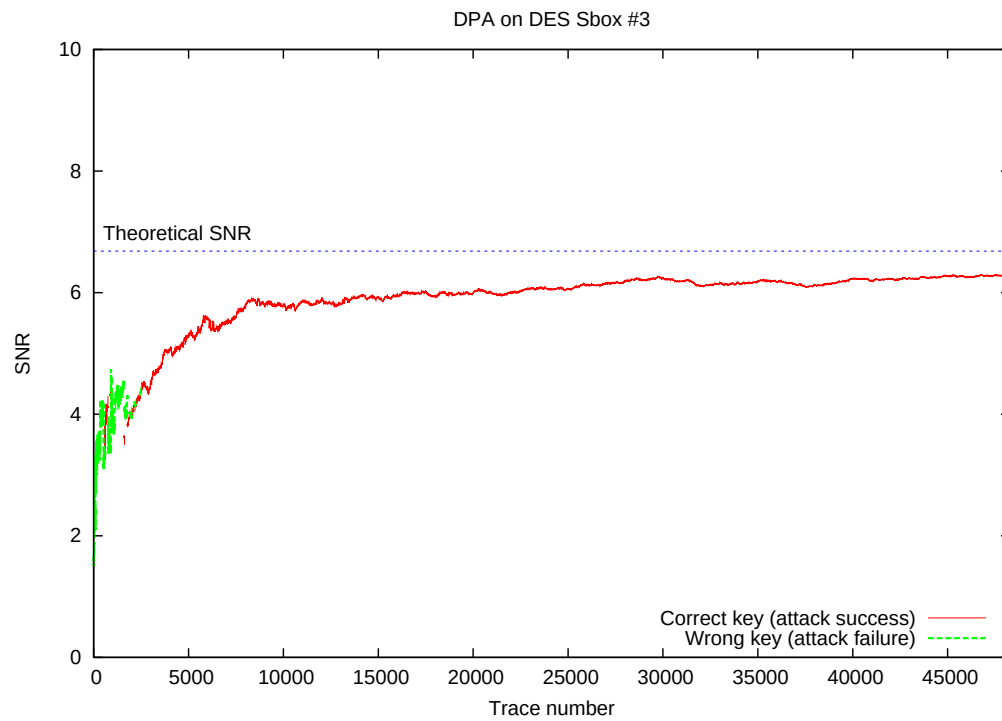


Figure B.11: SNR of the DPA on DES sbox #3.

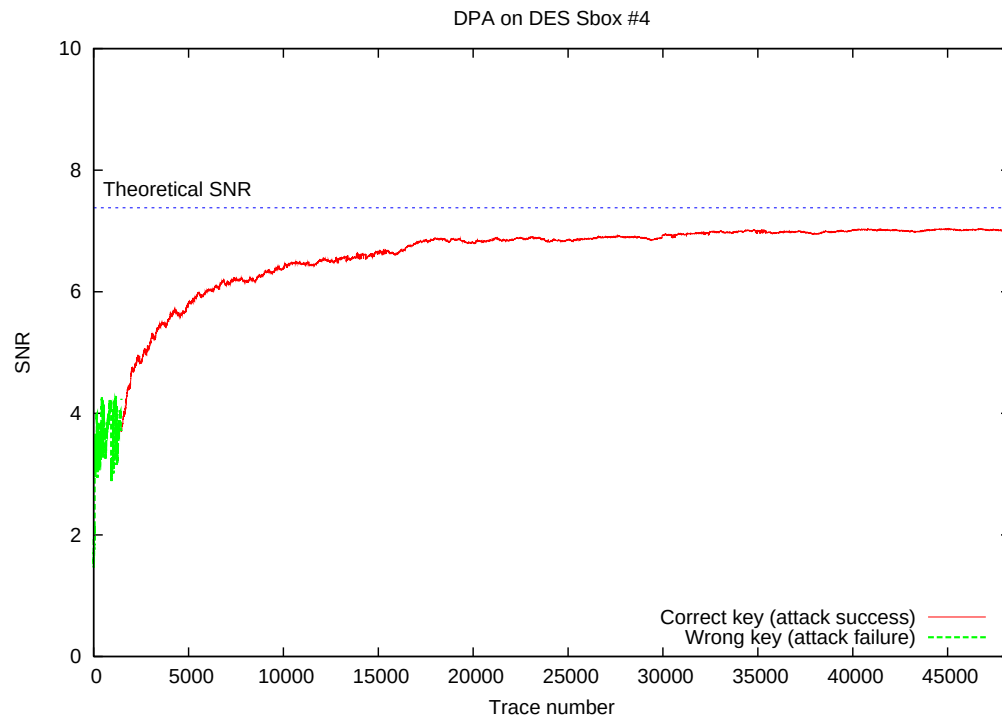


Figure B.12: SNR of the DPA on DES sbox #4.

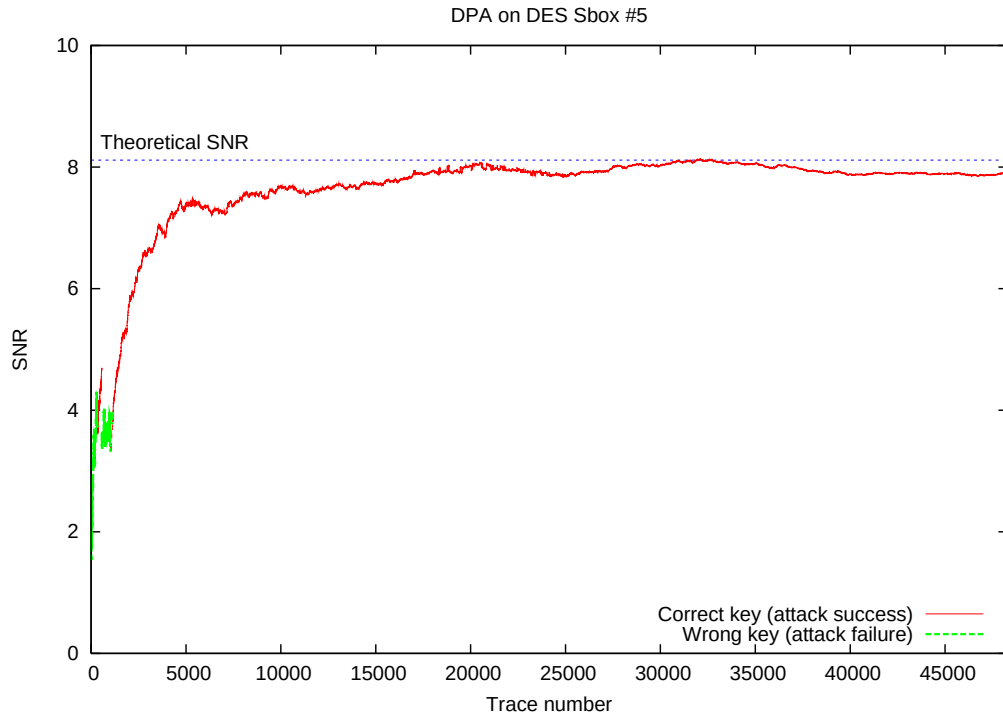


Figure B.13: SNR of the DPA on DES sbox #5.

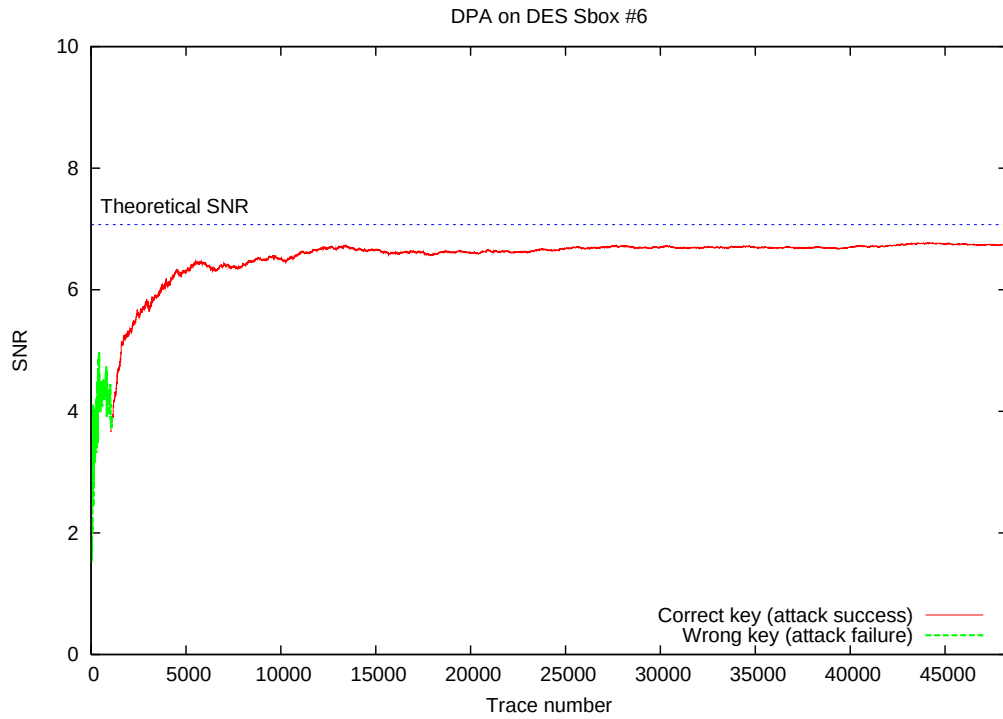


Figure B.14: SNR of the DPA on DES sbox #6.

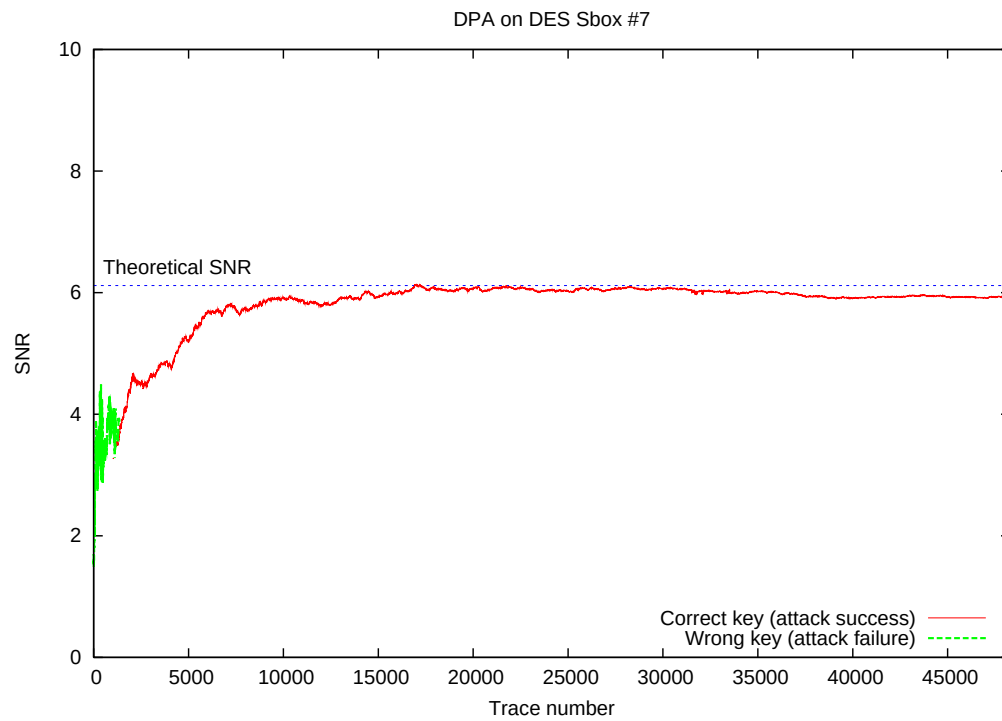


Figure B.15: SNR of the DPA on DES sbox #7.

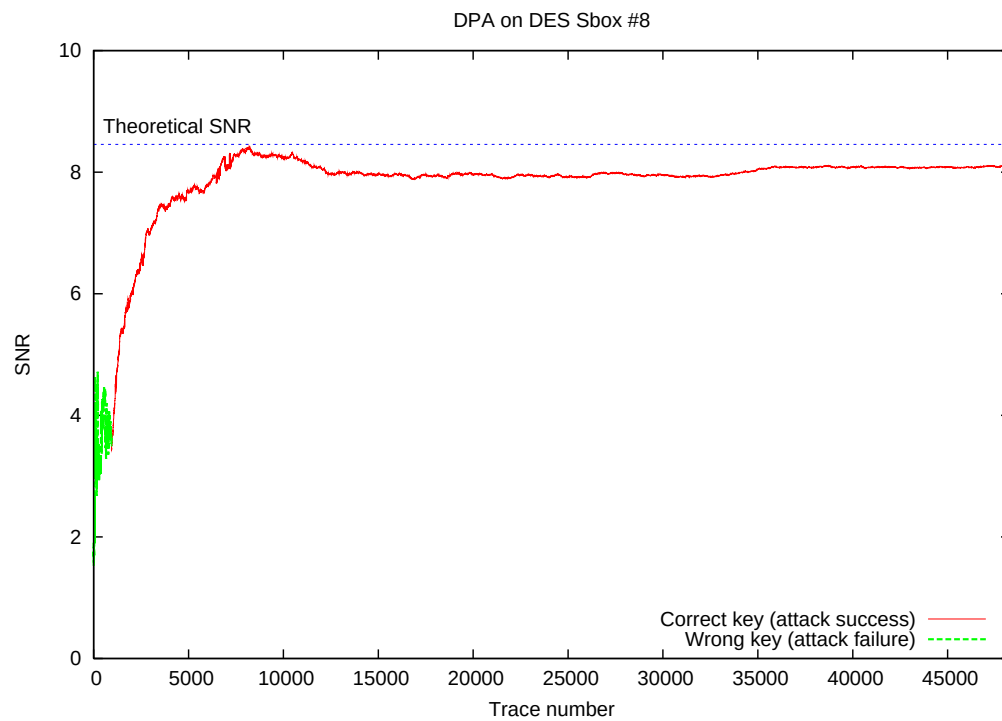


Figure B.16: SNR of the DPA on DES sbox #8.

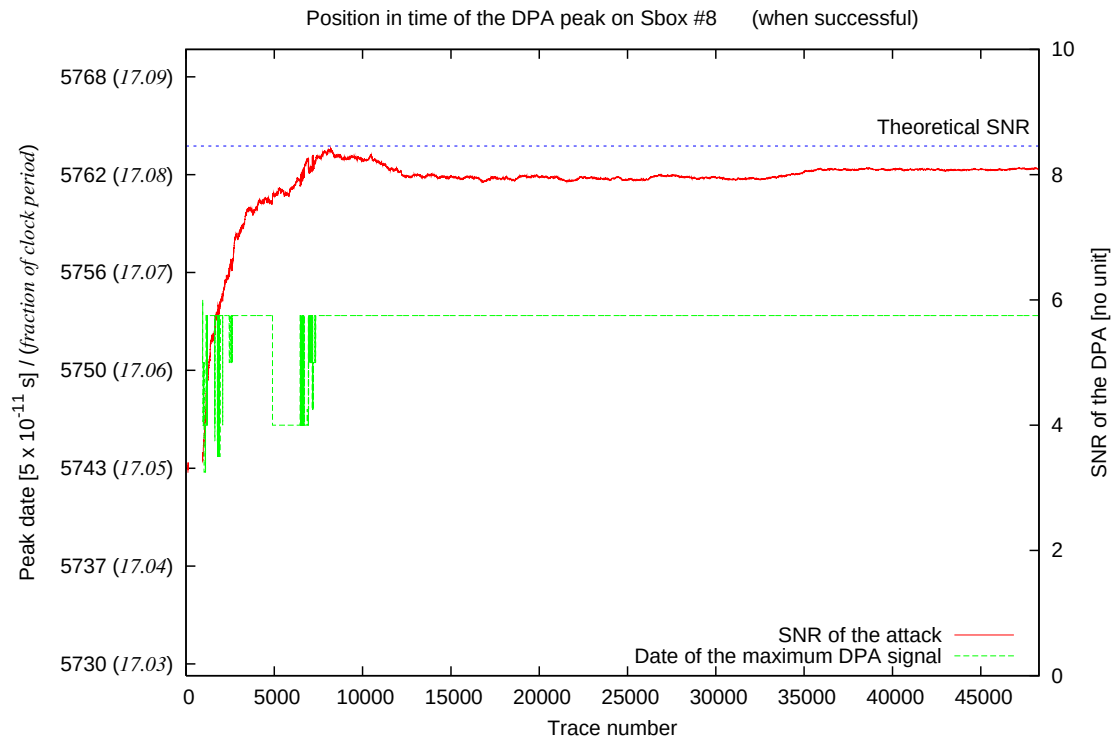


Figure B.17: Date selected for the SNR computation while doing the DPA on DES sbox #8.

Appendix C

Glossary

The acronyms specific to the topics approached in this document are listed in the following table.

Acronym	Signification
AES	Advanced Encryption Standard [72]
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design (<i>French acronym: CAO</i>)
CAO	Conception Assistée par Ordinateur (<i>English acronym: CAD</i>)
CISC	Complex Instruction Set Computer
CLB	Compound Logic Block
CMOS	Complementary MOS transistor
CMP	Circuit Multi-Projets (French broker in ICs & MEMS: http://cmp.imag.fr)
CPA	Correlation Power Analysis [111, 112]
CPU	Central Processing Unit
CTE	Common Timing Engine [23]
DES	Data Encryption Standard [71]
DFA	Differential Fault Attack [17] (particular type of FA)
DFF	D-type Flip-Flop (logical synchronous sample-and-memorize one-bit device)
DI	Delay Insensitive
DNF	Disjunctive Normal Form
DPA	Differential Power Analysis [54]
ECC	Elliptic Curve Cryptography
EMA	ElectroMagnetic Attack [37]
EMI	ElectroMagnetic Interference
FA	Fault Attack
FE	First Encounter, backend design product of Cadence [47]
FPGA	Field Programmable Gate Arrays
GNU	Gnu is Not Unix (Project website: http://www.gnu.org/)
GPIB	General Purpose Interface Bus, IEEE 488
HD, HW	Hamming Distance, Hamming Weight
IC	Integrated Circuit (typically an ASIC or an FPGA)
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers (http://www.ieee.org/)
IP	Initial Permutation, used in DES (Note that: $IP^{-1} \doteq FP$)
IP	Intellectual Property (<i>understand: “stand-alone hardware macro”</i>)

Continued on next page ...

Continued from previous page ...

IPA	Inferential Power Attack [32]
IPsec	Internet Protocol security, RFC 2401 [49]
IV	Initialization Vector
LE	Logic Element
LR	Left-Right, the 2×32 -bit state register of DES [71]
LuT	Look-up Table
MAC	Message Authentication Code
MD5	Message Digest #5, RFC 1321 (http://tools.ietf.org/html/rfc1321)
MOS	Metal-Oxide-Semiconductor (based on Silicon for most digital ICs)
MOSIS	MOS Implementation System (USA MPW company [4])
MPW	Multi-Project Wafer
NIST	National Institute of Standards and Technology (http://www.nist.gov/)
NP	Non-Polynomial, in complexity theory
OSI	Open Systems Interconnection [110]
PC	Personal Computer (trademark of IBM)
PCB	Printed Circuit Board
PKI	Public Key Infrastructure
P&R	Place-and-Route
QDI	Quasi-Delay Insensitive
RAM	Random Access Memory
RC5	Rivest Cipher #5 (RSAsecurity proprietary encryption algorithm)
RNG	Random Number Generator
ROM	Read-Only Memory
RSA	Rivest Shamir Adelman encryption/signature asymmetrical patented algorithm
RTL	Register Transfer Level (architectural notion)
RTZ	Return To Zero, <i>aka</i> Return To NULL in the context of QDI logic
Sbox	Substitution box, <i>aka</i> a vectorial Boolean function
SCA	Side-Channel Attack
SDF	Standard Delay Format [5], <i>IEEE standard #P1497</i>
SHA	Secure Hash Algorithm [69]
SI	Speed-Independent circuits (<i>see also</i> entries DI and QDI)
SNR	Signal-to-Noise Ratio (<i>refer to Eqn. (2.7) at page 32</i>)
SoC	System-on-Chip
SoI	Silicon-on-Insulator
SPA	Simple Power Analysis [54]
SPEF	Standard Parasitic Exchange Format (part of the 1481-1999 IEEE standard)
SPICE	Simulation Program with Integrated Circuit Emphasis [78]
SPN	Substitution – Permutation Network
TCG	Trusted Computing Group (https://www.trustedcomputinggroup.org/ , [6])
TCL	Tool Command Language (<i>see for instance</i> http://www.tcl.tk/)
TPM	Trusted Platform Module (<i>Refer to the entry “TCG”</i>)
T-RNG	True RNG (<i>i.e.</i> an RNG that is not ruled by any algorithm)
UML	Universal Modeling Language [8, 95]
USB	Universal Serial Bus (http://www.usb.org/)
VCI	Virtual Component Interface [108]
VHDL	VHSIC Hardware Description Language, <i>IEEE standard #1076-2000</i>
VHSIC	Very High Speed IC
VITAL	VHDL Initiative Towards ASIC Libraries [48], <i>IEEE standard #1076.4-2000</i>
WDDL	Wave Dynamic Differential Logic [105]
XOR	eXclusive OR, also denoted “ $\oplus$ ”

Bibliography

- [1] GNU/Electric CAD system. <http://www.gnu.org/software/electric/>.
- [2] ISO/IEC 7816, Smartcard Standard. ([Informal website](#)).
- [3] ITRS (International Technology Roadmap for Semiconductors) website. <http://public.itrs.net/>.
- [4] MOSIS (MOS Implementation System) website. <http://www.mosis.org/>.
- [5] Standard Delay Format (SDF) website. http://www.eda.org/sdf/sdf_3.0.pdf.
- [6] Trusted Computing Group (TCG). <http://www.trustedcomputinggroup.org/>.
- [7] IEEE Std 1481-1999, IEEE standard for integrated circuit (IC) delay and power calculation system, June 1999. [PDF \(IEEE Xplore\)](#).
- [8] “The Object Management Group (OMG) website”, 2006.
- [9] Alin Razafindraibe. *Analyse et Amélioration de la Logique Double Rail pour la Conception de Circuits Sécurisés*. PhD thesis, Université Montpellier II, November 2006. <http://papyrus.lirmm.fr/Document.htm&numrec=031994468917620> (french).
- [10] ATMEL. Datasheet – Triple Data Encryption Standard (TDES). March 2005. ([Online reference](#)).
- [11] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer’s apprentice guide to fault attacks. Cryptology ePrint Archive, Report 2004/100, 2004. <http://eprint.iacr.org/2004/100/>.
- [12] Charles H. Bennett. Invited conference on “Quantum Communication and Computation”. 15 february 2006, amphithéâtre Émeraude, at the CNRS [LTCL](#), [ENST](#) (Paris).
- [13] Charles H. Bennett and Gilles Brassard. Quantum Cryptography: Public Key Distribution and Coin Tossing. In *Proceedings of the International Conference on Computers, Systems, and Signal Processing*, 1984.
- [14] Benoît Chevallier-Mames and Mathieu Ciet and Marc Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004. <http://www.gemplus.com/smart/rd/publications/pdf/CCJ04ato.pdf>.
- [15] Guido Bertoni, Marco Macchetti, Luca Negri, and Pasqualina Fragneto. Power-Efficient ASIC Synthesis of Cryptographic S-Boxes. In *ACM Great Lakes Symposium on VLSI*, pages 277–281, april 2004. Boston, MA, USA. ([PDF available from ACM – http://portal.acm.org/](#)).

- [16] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
- [17] Eli Biham and Adi Shamir. Differential Fault analysis on secret key cryptosystems. In *Proc. of CRYPTO'97*, volume 1294, pages pp 513–525, 1997.
- [18] Alex Biryukov and Adi Shamir. Structural Cryptanalysis of SASAS. In *Proceedings of Eurocrypt 2001*, volume LNCS 2045, pages 394–405, 2001.
- [19] E. Boros and P.L. Hammer. Pseudo-Boolean Optimization. *Discrete Applied Mathematics*, 123((1-3)):155–225, 2002.
- [20] G.F. Bouesse, M. Renaudin, S. Dumont, and F. Germain. DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement. In *Proceedings of DATE'05*, pages pp 424–429, March 2005. Munich, Germany.
- [21] G.F. Bouesse, M. Renaudin, B. Robisson, E. Beigné, P.-Y. Liardet, S. Prevosto, and J. Sonzogni. DPA on Quasi Delay Insensitive Asynchronous Circuits: Concrete Results. In *XIX Conference on Design of Circuits and Integrated Systems, Proceedings of DCIS'04*, 24–26 Nov 2004. Bordeaux, France ([PDF](#)).
- [22] Bruce Schneier. *Applied Cryptography*. 1996. John Wiley & Sons, ISBN 0-471-12845-7.
- [23] Cadence. Delay Calculation Algorithm Guide, june 2002. Product SPR50, [ct_alg.pdf](#).
- [24] Claude Carlet. On Highly Nonlinear S-Boxes and Their Inability to Thwart DPA Attacks. pages 49–62. INDOCRYPT 2005 (LNCS 3797), december 2005. Bangalore, India. ([PDF](#) on [SpringerLink](#); Complete version on [IACR ePrint](#)).
- [25] S. Chari, J.R. Rao, and P. Rohatgi. Template Attacks. In *CHES*, volume 2523 of *Lecture Notes in Computer Science*, August 2002. ISBN: 3-540-00409-2.
- [26] Nicolas T. Courtois, Guilhem Castagnos, and Louis Goubin. What do DES S-boxes Say to Each Other? Cryptology ePrint Archive, Report 2003/184, 2003. <http://eprint.iacr.org/2003/184>.
- [27] “Distributed” project website. <http://www.distributed.org/>.
- [28] Electronic Frontier Foundation (EFF – <http://www EFF.org/>). *Secrets of Encryption Research, Wiretap Politics & Chip Design*. July 1998. ISBN: 1-56592-520-3.
- [29] Elisabeth Oswald. *On Side-Channel Attacks and the Application of Algorithmic Countermeasures*. PhD thesis, IAIK, may 2003. <http://www.iaik.tu-graz.ac.at/aboutus/people/oswald/papers/PhD.pdf>.
- [30] Eric Young, <eay@cryptsoft.com>. DES ASM and C implementation in openssl (file `crypto/des/des.h`), 1995–1997. ([Source code](#)).
- [31] Fabien Germain. *Towards cryptographic security using dedicated integrated circuits design methodologies*. PhD thesis, École Polytechnique, June 2006. <http://www.imprimerie.polytechnique.fr/Theses/Files/Germain.pdf> (french).
- [32] Paul N. Fahn and Peter K. Pearson. IPA: A New Class of Power Attacks. In *Proc. of CHES*, volume LNCS 1717, pages 173–186, 1999. <http://link.springer.de/link/service/series/0558/bibs/1717/17170173.htm>.

- [33] Karl Fant and Scott Brandt. NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis. In *International Conference on Application Specific Systems, Architectures, and Processors (ASAP 96)*, pages 261–273, August 1996. <http://www.theseusresearch.com/Downloads/NCL.PDF>.
- [34] Florent Chabaud and Serge Vaudenay. Links between Differential and Linear Cryptanalysis. In *Proc. of Eurocrypt'94*, volume 950, pages 356–365, 1995. Springer-Verlag, (PDF).
- [35] Edward Fredkin and Tommaso Toffoli. Conservative Logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.
- [36] G. Rouvroy and F.-X. Standaert and J.-J. Quisquater and J.-D. Legat. Efficient Use of FPGAs for Implementations of DES and Its Experimental Linear Cryptanalysis. *IEEE Transactions on Computers*, 52(4), April 2003.
- [37] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In *Proceedings of CHES'01*, volume 2162 of *LNCS*, pages pp 251–261. Springer, May 2001. <http://www.gemplus.com/smart/rd/publications/ps/GM001ema.ps.gz>.
- [38] Ghislain Fraidy Bouesse. *Contribution à la Conception de Circuits Intégrés Sécurisés : l'Alternative Asynchrone*. PhD thesis, Institut National Polytechnique de Grenoble (INPG) – TIMA, december 2005. http://tima.imag.fr/publications/files/th/csd_221.pdf.
- [39] Gilles Piret. A Note on the Plaintexts Choice in Power Analysis Attacks. Technical report, November 2005. <http://www.di.ens.fr/~piret/publ/power.pdf>.
- [40] Gilles Piret and Jean-Jacques Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In *"CHES'03"*, volume *LNCS 2779*, pages 77–88, 2003. (Online PDF version).
- [41] L. Goubin and J. Patarin. DES and Differential Power Analysis (The “Duplication” Method). In *Proceedings of CHES'99*, volume 1717 of *LNCS*, pages pp 158–172. Springer, August 1999.
- [42] S. Guilley, Ph. Hoogvorst, Y. Mathieu, R. Pacalet, and J. Provost. CMOS Structures Suitable for Secured Hardware. In *Proceedings of DATE'04*, pages pp 1414–1415, February 2004.
- [43] S. Guilley and R. Pacalet. SoC Security: a War against Side-Channels. *Annals of the Telecommunications*, 59(7–8):998–1009, july–august 2004. (Abstract – Full Paper).
- [44] Helion Technology. Datasheet – High Performance DES and Triple DES core for ASIC. 2003. (Online reference).
- [45] Horst Feistel. Cryptography and Computer Privacy. *Scientific American*, pages 15–23, May 1973. (Online PDF version).
- [46] Synopsys. Liberty Vol. 1 & 2, dec 2003. “liberty.pdf”, available from Synopsys “tap-in” program website: <http://www.synopsys.com/partners/tapin/>.
- [47] Cadence. First Encounter Silicon Virtual Prototyping, Encounter® digital IC design platform,. http://www.cadence.com/products/digital_ic/first_encounter/.
- [48] IEC 61691-5:2004, IEEE standard #1076.4-2000. *VITAL (VHDL Initiative Towards ASIC Libraries) ASIC (application specific integrated circuit) modeling specification*.

- [49] “IPsec”. Security Architecture for the Internet Protocol, <http://rfc.net/rfc2401.html> or <http://www.ietf.org/rfc/rfc2401.txt>.
- [50] J.-L. Danger and S. Guilley and Ph. Matherat and Y. Mathieu and L. Naviner and A. Polti and J. Provost. “Électronique Numérique Intégrée”. Cours de l’École Nationale Supérieure des Télécommunications, Paris, 2006. <http://www.comelec.enst.fr/tpsp/eni/poly/eni.pdf>.
- [51] Jean-Jacques Quisquater and François-Xavier Standaert. Exhaustive Key Search of the DES: Updates and Refinements. February 2005. <http://www.ruhr-uni-bochum.de/itsc/tanja/SHARCS/>.
- [52] Joe Kilian and Phillip Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *Journal of Cryptology*, 14(1):17–35, 2001.
- [53] P. Kocher, J. Jaffe, and B. Jun. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of CRYPTO’96*, volume 1109 of *LNCS*, pages pp 104–113. Springer, 1996. Springer-Verlag, (PDF).
- [54] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis: Leaking Secrets. In *Proceedings of CRYPTO’99*, volume 1666 of *LNCS*, pages pp 388–397. Springer, 1999. Springer-Verlag, (PDF).
- [55] Cédric Lauradoux and Ronan Keryell. CryptoPage-2 : un processeur sécurisé contre le jeu. In *Proc. of RENPAR’15 / CFSE’3 / SympAAA’2003*, October 2003. <http://www.lit.enstb.org/~keryell/publications/conf/2003/SympAAA/article.pdf>.
- [56] LEF/DEF parsers, website. <http://openeda.si2.org/projects/lefdef/> or <http://www.cadence.com/partners/languages/languages.aspx>.
- [57] Régis Leveugle. Automatic modifications of high level VHDL descriptions for fault detection or tolerance. In *Proceedings of DATE’02*, pages pp 837–841, March 2002.
- [58] M. Akkar and C. Giraud. An Implementation of DES and AES secure against Some Attacks. In Springer-Verlag, editor, *Proc. of CHES’01*, number 2162, pages 309–318, 2001.
- [59] M. Matsui. Linear cryptanalysis method for DES cipher. In *Proceedings Eurocrypt’93*, T. Helleseht, Ed., Springer-Verlag, (LNCS 765):386–397, 1994.
- [60] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. october 1996. CRC Press, ISBN: 0-8493-8523-7, 816 pages, <http://www.cacr.math.uwaterloo.ca/hac/>.
- [61] T.S. Messerges, E.A. Dabbish, and R.H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard Technology*, pages pp 151–162, May 1999.
- [62] David Molnar, Matt Piotrowski, David Schultz, and David Wagner. The Program Counter Security Model: Automatic Detection and Removal of Control-Flow Side Channel Attacks. 2005. [Cryptology ePrint Archive](#), report 2005/368.
- [63] J. H. Moore and G. J. Simmons. Cycle Structure of the DES with Weak and Semi-Weak Keys. In *Proceedings of CRYPTO*, number LNCS 263, pages 3–32, August 1986. Springer-Verlag, Berlin, 1987; Santa Barbara, USA.

- [64] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor. Improving Smart Card Security using Self-timed Circuits. In *Proceedings of ASYNC'02*, pages pp 211–218., April 2002. Manchester, United Kingdom.
- [65] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, 2004. (3rd Edition), ISBN: 0-321-14901-7, <http://www.cmosvlsi.com/>.
- [66] Network of Excellence **ECRYPT** virtual lab **VAMPIRE** (virtual application and implementation research laboratory). The “Side-Channel Cryptanalysis” Lounge , 2006. http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html.
- [67] NIST/ITL/CSD. DES Modes of Operation, December 1980. ([Online reference](#)).
- [68] NIST/ITL/CSD. FIPS PUB 74: Guidelines for implementing and using the NBS Data Encryption Standard, April 1981. <http://www.itl.nist.gov/fipspubs/fip74.htm>.
- [69] NIST/ITL/CSD. Secure Hash Standard. FIPS PUB 180-1, 1993. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- [70] NIST/ITL/CSD. Modes of Operation Validation System (MOVS): Requirements and Procedures. February 1998. ([Online reference](#)).
- [71] NIST/ITL/CSD. Data Encryption Standard. FIPS PUB 46-3, October 1999. <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>.
- [72] NIST/ITL/CSD. FIPS PUB 197: Advanced Encryption Standard (AES), November 2001. ([Online reference](#)).
- [73] Renaud Pacalet. “Security of Security Hardware” graduate course at **Institut Eurecom**. <http://soc.eurecom.fr/crypto>.
- [74] Renaud Pacalet and Sylvain Guilley. **CMP** Annual Report 2005. (Online PDF versions: [web](#) / [local](#)).
- [75] Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Special Issue of Integration, The VLSI Journal*: “Embedded Cryptographic Hardware”, 2006. ([Online PDF](#)).
- [76] Philippe Matherat and Marc-Thierry Jaeckel. Dissipation logique des implémentations d’automates — Dissipation du calcul. *Technique et Science Informatique*, 15(8):1079–1104, october 1996.
French version: <http://www.comelec.enst.fr/~matherat/publications/tsi96/>;
English version: <http://fr.arxiv.org/abs/quant-ph/9805018>.
- [77] Emmanuel Prouff. DPA Attacks and S-Boxes. pages 424–441. **FSE 2005** (LNCS 3557), february 2005. Paris, France. (Edited by **Springer-Verlag**).
- [78] Jan M. Rabaey. SPICE (language & simulator) website. <http://bwrc.eecs.berkeley.edu/Courses/IcBook/SPICE/>.
- [79] Ralph Merkle and Martin Hellman. On the Security of Multiple Encryption. *Communications of the ACM*, 24(7):465–467, July 1981.

- [80] A. Razafindraibe, M. Robert, and P. Maurine. Asynchronous Dual Rail Cells to Secure Cryptosystems against Side Channel Attacks. In *Proc. of SAME 2005 forum, 8th edition*. Sophia Antipolis, France, October 6th 2005.
- [81] Richard Clayton and Mike Bond. Experience Using a Low-Cost FPGA Design to Crack DES Keys. In *Cryptographic Hardware and Embedded Systems (CHES'02)*, volume LNCS 2523, pages 579–592, Aug 2002.
- [82] Vincent Rijmen. Efficient Implementation of the Rijndael S-box. (Short note in [PDF](#)).
- [83] Ronald L. Rivest. Message Digest 5. RFC 1321.
<http://theory.lcs.mit.edu/~rivest/Rivest-MD5.txt>.
- [84] P. Rogaway. The Security of DESX, 1996. RSA Laboratories Cryptobytes, citeseer.ist.psu.edu/rogaway96security.html.
- [85] Ross J. Anderson. Serpent website (former candidate to the AES), 1999.
<http://www.cl.cam.ac.uk/~rja14/serpent.html>.
- [86] Régis Bévan. *Évaluation statistique et sécurité des cartes à puce. Évaluation d'attaques DPA évoluées*. PhD thesis, (french). Université Paris 11 & École Nationale Supérieure d'Électricité ([Supélec](#)), April 2004.
- [87] S. Chaudhuri and J.-L. Danger and S. Guilley and Ph. Hoogvorst. FASE: An Open Run-Time Reconfigurable FPGA Architecture for Tamper-Resistant and Secure Embedded Systems. In *3rd international conference on reconfigurable computing and FPGAs (ReConFig 2006)*, September 2006. San Luís Potosí, México, ([Online PDF](#)).
- [88] S. Guilley and Ph. Hoogvorst. The Proof by $2^M - 1$: a Low-Cost Method to Check Arithmetic Computations. In *SEC 2005*, volume IFIP 181, pages pp. 589–600, May 2005. Makuhari-Messe, Chiba, Japan. ([PDF](#)).
- [89] S. Guilley and Ph. Hoogvorst and R. Pacalet. Differential Power Analysis Model and some Results. In *Proceedings of WCC/CARDIS'04*, pages pp 127–142, August 2004. Toulouse, France.
- [90] Takashi Satoh, Tetsu Iwata, and Kaoru Kurosawa. On Cryptographically Secure Vectorial Boolean Functions. In *Proc. of Asiacrypt'99*, volume 1716, pages 20–28, 1999. Springer Verlag.
- [91] A. Schubert, R. Jährgig, and W. Anheier. Cryptography Reuse Library. In *Forum on Design Languages (FDL 99)*. Lyon, France, August 1999.
- [92] Sci-worx. Datasheet – DES / Triple DES (High Performance). ([Online reference](#)).
- [93] M. Shams, J.C. Ebergen, and M.I. Elmasry. Modeling and comparing CMOS implementations of the C-element. *IEEE Transactions on VLSI Systems*, 6(4):563–567, December 1998.
- [94] Simon Moore and Ross Anderson and Robert Mullins and George Taylor and Jacques J.A. Fournier. Balanced Self-Checking Asynchronous Logic for Smart Card Applications. *Journal of Microprocessors and Microsystems*, 27(9):421–430, October 2003.
<http://www.cl.cam.ac.uk/~swm11/research/papers/micromicro2003.pdf>.
- [95] Sinan Si Alhir. “[Learning UML](#)”, July 2003. ISBN: 0-596-00344-7.
- [96] D. Sokolov, J. Murphy, and A. Bystrov. Improving the Security of Dual-Rail Circuits. In *Proceedings of CHES'04*, LNCS, pages pp 282–297. Springer, Aug 2004.

- [97] Stefan Mangard and Elisabeth Oswald and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smartcards*. Springer, December 2006. ISBN 0-387-30857-1, <http://www.dpabook.org/>.
- [98] E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. The AEGIS processor architecture for tamper evident and tamper resistant processing, 2003. Technical Report LCS-TM-461, Massachusetts Institute of Technology.
- [99] Ivan E. Sutherland. Micropipelines (*Turing award*). *Communications of the ACM*, 32(6):720–738, June 1989.
- [100] Daisuke Suzuki and Minoru Saeki. Security evaluation of dpa countermeasures using dual-rail pre-charge logic style. In *CHES*, pages 255–269, 2006. http://dx.doi.org/10.1007/11894063_21.
- [101] Sylvain Guilley and Philippe Hoogvorst and Renaud Pacalet. A Fast Pipelined Multi-Mode DES Architecture Operating in IP Representation. *Integration, The VLSI Journal*, to appear in 2006. DOI: 10.1016/j.vlsi.2006.06.004.
- [102] Sylvain Guilley, Philippe Hoogvorst, Yves Mathieu and Renaud Pacalet. The “Backend Duplication” Method. In *CHES 2005*, volume LNCS 3659, pages 383–397. Springer. August 29th – September 1st, Edinburgh, Scotland, UK ([Online presentation](#)).
- [103] K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *Proceedings of ESSCIRC’02*, pages pp 403–406, September 2002.
- [104] K. Tiri and I. Verbauwhede. Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In LNCS, editor, *Proceedings of CHES’03*, volume 2779 of LNCS, pages pp 125–136. Springer, September 2003.
- [105] K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *Proceedings of DATE’04*, pages pp 246–251, February 2004.
- [106] K. Tiri and I. Verbauwhede. Place and Route for Secure Standard Cell Design. In *Proceedings of CARDIS’04*, pages pp 143–158, August 2004.
- [107] Tommaso Toffoli. Bicontinuous Extensions of Invertible Combinatorial Functions. *Mathematical Systems Theory*, 14:13–23, 1981.
- [108] VSI Alliance. On-Chip Bus Development Working Group. Virtual Component Interface Standard Version 2 (OCB 2 2.0), April 2001. <http://www.vsia.org/>.
- [109] Z.C. Yu, S.B. Furber, and L.A. Plana. An investigation into the Security of Self-timed Circuits. In *Proc. of Async’03*, May 2003. Vancouvers, Canada.
- [110] H. Zimmerman. OSI reference model — The OSI model for architectures for open systems interconnection. *IEEE Transactions on Communications*, 28:425–432, April 1980.
- [111] Éric Brier, Christophe Clavier, and Francis Olivier. Optimal statistical power analysis. Cryptology ePrint Archive, Report 2003/152, 2003. <http://eprint.iacr.org/>.
- [112] Éric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. *Proc. of CHES’04*, LNCS 3156:16–29, August 11–13 2004. ISSN: 0302-9743; ISBN: 3-540-22666-4; DOI: 10.1007/b99451; Cambridge, MA, USA.