Random Active Shield

Sébastien Briais, Jean-Michel Cioranesco, Jean-Luc Danger, Sylvain Guilley, David Naccache, Thibault Porteboeuf

Random Active Shield

Sébastien BRIAIS[1], Jean-Michel CIORANESCO[2,3], Jean-Luc DANGER[1,4],
Sylvain GUILLEY[1,4], David NACCACHE[3,5] and Thibault PORTEBOEUF[1]

[1]*Secure-IC S.A.S., 37/39 rue Dareau, 75 014 Paris, France* and *80 avenue des Buttes de Coësmes, 35 700 Rennes, France.*
{`sebastien.briais, sylvain.guilley, jean-luc.danger, thibault.porteboeuf`}`@secure-ic.com`

[2]*Altis Semiconductor, 224 Boulevard John Kennedy, 91 100 Corbeil-Essonnes, France.*
`jean-michel.cioranesco@altissemiconductor.com`

[3]*Sorbonne Universités – Université Paris II, 12 place du Panthéon, 75 231, Paris Cedex 05, France.*
`jean-michel.cioranesco@etudiants.u-paris2.fr`

[4]*Institut MINES-TELECOM, TELECOM-ParisTech, CNRS LTCI (UMR 5141),*
46 rue Barrault, 75 634 Paris Cedex 13, France and *37/39 rue Dareau, 75 014 Paris, France.*
{`jean-luc.danger, sylvain.guilley`}`@telecom-paristech.fr`

[5]*École normale supérieure, Département d'informatique, 45, rue d'Ulm, 75 230, Paris Cedex 05, France.*
`david.naccache@ens.fr`

***Abstract*—Recently, some active shielding techniques have been broken (*e.g.* by FlyLogic). The caveat is that their geometry is easy to guess, and thus they can be bypassed with an affordable price. This paper has two contributions. First of all, it provides a definition of the objectives of shielding, which is seldom found in publicly available sources. Notably, we precise the expected functionality, but also the constraints it must meet to be both manufacturable and secure. Second, we propose an innovative solution based on random shielding. The goal of this shielding is to make the geometry of the shield difficult to recognize, thereby making the "identification" phase of the attack harder than in previous schemes. Also, a proof of the shielding existence for two layers of metal is provided, which guarantees that the generation of the layout will succeed. Finally, we provide real tests of the shield generation algorithm, that show it is computationally tractable even for large areas to protect.**

***Keywords*-Active shield, spaghetti routing, harder identification phase, traveling salesman problem (TSP), genetic algorithms.**

I. Introduction

Cryptographic circuits must be protected against attacks that aim at extracting the information they conceal. Probing is a popular way to read or write data using a port (the probe tip) normally unavailable to the attacker. Thus, shielding protections were devised and implemented on top of most secure chips.

At the early ages of smartcards, the chips' integrated nature was already a good protection. It was believed that it is *a priori* hard for average hackers to access the contents of integrated circuits. Progressively, test tools such as probing stations, normally used to debug live circuits, became increasingly available. Naturally, these also turned to be relevant attack tools. In the meantime, more advanced CMOS technologies were being deployed, feature size decreased and the number of interconnect layers increased. This deterred probing, and was indeed taken advantage of by designers to further obfuscate circuit using random placement and routing of sensitive gates. Thanks to recent technological advances, a revived interest in probing attacks is witnessed since 2010. Especially, these have been aided by the increased popularity of the possibility to draw artificial pads that conduct directly into the inner parts of the circuit, thanks to focused ion beam (FIB) tools. Consequently, circuits shielding is thus still mandatory, and especially relevant nowadays.

This article is structured as follows. Active shield techniques and attacks are briefly surveyed in Sec. II. The specification of feasible and secure shielding geometries is given in Sec. III. Stemming from this formalization, a new solution, aimed at avoiding trivial shield reconstruction, is explained in Sec. IV, and conclusions and perspectives are drawn in Sec. V.

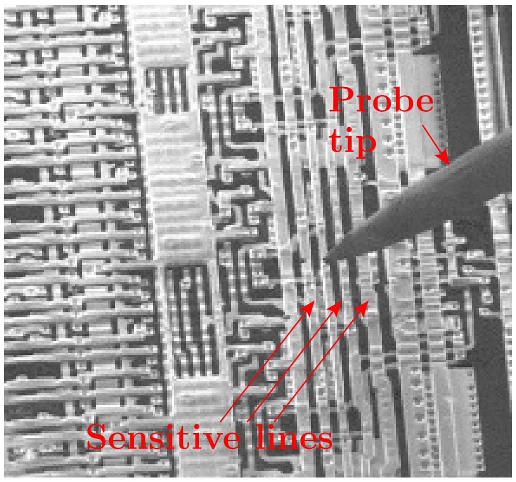


Figure 1. Probing of a circuit thanks to prober tip, to read or force sensitive variables (courtesy of [4], Fig. 4.1 of §4.2. at page 31).

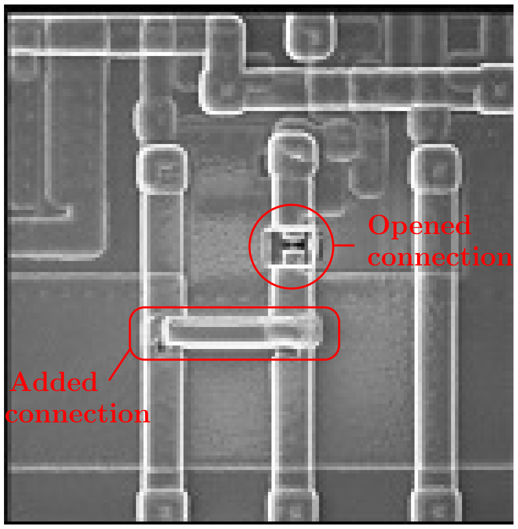


Figure 2. Edition of a circuit thanks to a FIB, in a view to unlock the access to a memory (courtesy of [4], Fig. 4.2 of §4.2. at page 31).

II. Overview of Shielding

The goal of shielding is to prevent attacks that consist in:

- either placing a probe on a resource (wire, gate, memory cell), for a subsequent probing (read and/or inject data into the device) during execution;
- or modifying a chip (using a FIB), and then running it.

The first attack typically allows to spy data on a bus, change access rights during memory writing, or alter opcodes read from program memory (Fig. 1). The second attack can be used to unlock resources (Fig. 2).

Conversely, shielding does not protect against reverse-engineering using methods such as layout recognition algorithms [11], [8]. While optical imaging allows to identify interconnect lines and gates functions, it usually fails to read non-volatile memory (NVM) contents (*e.g.* EEPROM cells). So, it is a safe practice to store keys in NVM memory.

Thus, the general principle of shielding is to cover a sensitive area by metal lines, meant to detect intrusions. Secure logic is thus sandwiched between the shield (top) and the substrate (bottom)[1]. Backside attacks are more chancy, since the layout does not clearly stand out. Furthermore, some encapsulation techniques in the module do not allow an accurate backside probing. Also, technologies such as SOI (Silicon on Insulator) are expected to forbid backside attacks. as well as other means of shield, like 3D canaries [3]. The kind of protection we thus consider is sketched in Fig. 3. The sensitive circuit is covered by metal line segments that guarantee the circuit is not functional if the shield is tampered with.

Passive shielding uses an analogue shield integrity measurement. For instance, the capacitive load of a line can serve as a signature. Passive shielding can however be defeated because it must tolerate some variations on the quantity being monitored. Thus, digital shielding (called active shielding) can be preferred. This consists in injecting random sequences of bits in a topmost metal circuit and checking that they arrive unaltered after their journey.

Nowadays threats are the attack of the active shield with the FIB. This protection can be defeated if the

[1]The optical PUFs [12] make use of the same strategy; they consist of a transparent material containing randomly distributed scattering particles allowing to deviate the laser light. Nonetheless, this optical PUF technology requires light sources and detectors, and the depositions of specific materials. It is thus not compatible with mainstream CMOS processes.

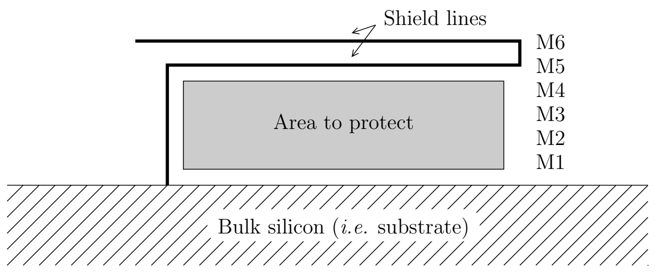


Figure 3. General structure of a shield (sagittal view).

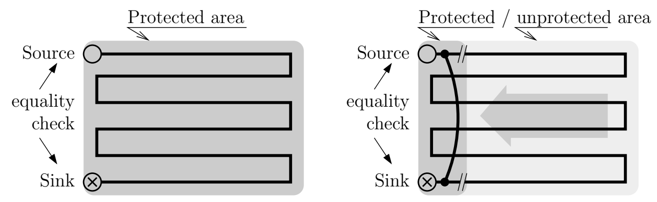


Figure 4. Area protected by a snake active shield (*left*), and shrunk protected area (*right*) by shield extension reduction (with cuts // and connections • introduced by FIB), at constant functionality (view of the top of the shield).

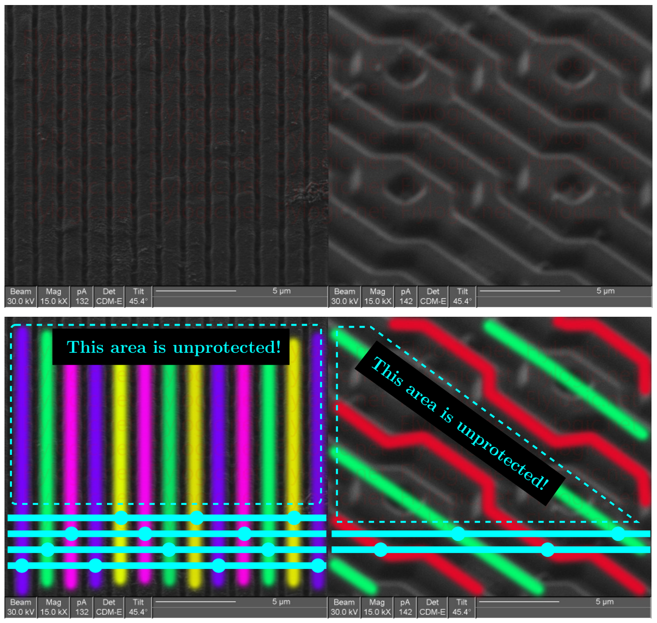


Figure 5. Zoom at 15,000 magnification of shield structures by Infineon (*left*) and STMicroelectronics (*right*). On the bottom annotated picture, equipotential lines are underlined with the same color. [Source: [10]]. The *rerouting attack* principle is illustrated in cyan superimposed comments.

meaning of the lines is disclosed, since their geometry can be changed while keeping the functionality invariant (Fig. 4). We refer to this kind of alteration by the term *shield rerouting attack*.

In practice, the identification of the identical lines is more complex than in the case of the single serpentine of Fig. 4. But, it is often possible to recognize the equipotential lines with well structured shields (Fig. 5).

III. Requirements of Shielding

Little information is generally available publicly about shielding specifications. The main reason is that this topic is usually disregarded by the scientific community. As a matter of fact, researchers preferably look for other "secure by design" protections, or consider that if an adversary has the power to probe lines, then most efforts to attempt to hamper him will only delay (but not prevent) a successful attack. The industry usually keeps the countermeasures secret, since, from a regulation point of view, it improves the attacks quotations (in terms of Common Criteria – CC [1]), and actually decreases the chance of a real-world attack once the product is on the field. Nonetheless, some companies patent shielding ideas, *e.g.* analog passive [7] or digital active [2] shield structures. Or we can also come across commercial brochures about active shielding [6], that explain some performance figures but not the protection rationale. Eventually, pirates sometimes disclose attacks, *e.g.* Tarnovsky (from FlyLogic [9]) on the Infineon SLE66 [10].

From this limited state-of-the-art, we nevertheless see that shielding is an industrial requirement and a target of attacks. Here, we intend to explain the specification of shielding and provide a rigorous ground for design practices. The two main challenges when devising a shield are manufacturability and security. Cost and power consumption efficiencies are other constraints, but it appears in practice that shielding uses only scarce resources (compared to the principal factors for area and power, that are memory and their accesses, and the IO for power). We detail both aspects in the following subsections.

A. Manufacturability Requirements for the Shielding

The shield must comply with some design rules checks (DRCs), that are described in the following subsections. For the sake of illustration, we consider a 0.13 μm technology from STMicroelectronics, with 6 levels of metal. They are called M1 to M6, and can be connected respectively by the vias V12, V23,

. . . , V56. We recall that modern routing practices [13] consist in aligning the metal wires on a grid, that coincides with the possible positions for the vias. Thus the connectivity in the horizontal 2D plane is achieved by metal segments, whereas vertically, between Mi and Mj (where $j = i \pm 1$), it is achieved by a via Vij.

1) Metal extension beyond a via at end of lines: The metal lines cannot stop dead after a via: an extension is required by the design rules. Therefore, either one routing site is skipped after each via, or the width of the wire is increased to absorb the extension. These two strategies are illustrated in Fig. 6. Here is a more accurate description of it:

(a) The extension after a via is a mandatory design rule, that makes up for possible masks misalignments during the fabrication process.
(b) The extension forbids to use all the possible slots available for vias.
(c) One solution, for instance used by automatic routers, is to forbid the use of the via next to the extension.
(d) Another solution is to make an abstraction of this extension by incorporating it into a new site placement grid (depicted in dotted fat gray lines in the figure).

2) Metal maximal parallel run length: Another design rule to consider is the maximum parallel run length. This rule aims at preventing two adjacent lines from being merged during fabrication. Thus the lines must be constrained to a given length. To respect this rule, some sites for M6 can be removed.

3) Density considerations: Lastly, the density rules must be verified. They state that the density must be neither too low nor too high, otherwise the chemical-mechanical planarization process will not manage to flatten the layer just deposited. However, by design, we are not concerned by this rule. Indeed, by using minimally (or near minimally) sized metal lines, we cannot reach the upper bound, due to the space between the wires. And as we wish to have most sites populated, we do not fall into the minimal density rule either.

4) Antennae rules check: During fabrication, the wires collect ions (charged particles) from plasmas used in chemical vapor depositions fabrication stages. These charges can accumulate at the CMOS transistor gates and irreversibly damage them by perforating the oxide. Therefore, it is required that a maximal area of metal is exposed on transistor gates while not being connected to a drain. This situation is, by design, avoided, as the shield makes many zigzags (which coincides with the silicon founder recommended advice to reduce the antenna effect) between levels of metals, thus preventing large area wires to be connected directly on a few transistor gates.

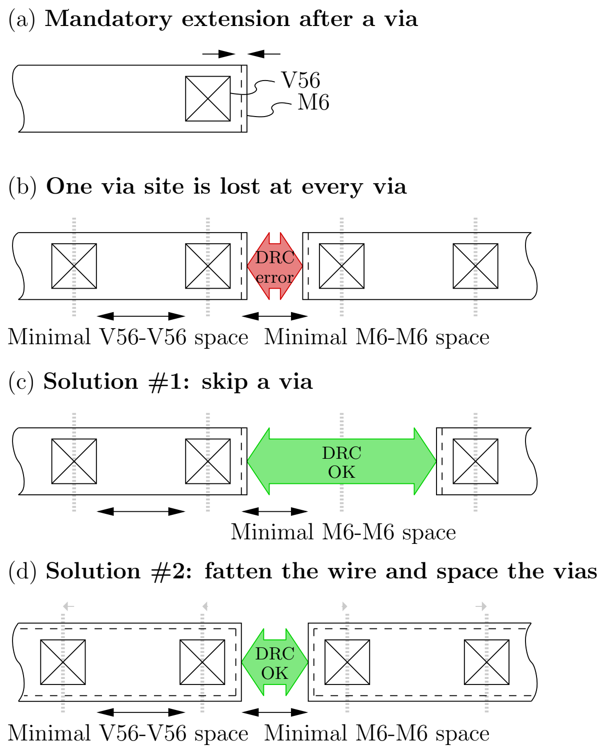

(a) Mandatory extension after a via
V56
M6
(b) One via site is lost at every via
DRC error
Minimal V56-V56 space
Minimal M6-M6 space
(c) Solution #1: skip a via
DRC OK
Minimal M6-M6 space
(d) Solution #2: fatten the wire and space the vias
DRC OK
Minimal V56-V56 space
Minimal M6-M6 space

Figure 6. Management of metal line extension beyond via end of line (extension).

B. Security Requirements for the Shielding

The shield feature size must be as small as possible, since FIBs have an edition capability of greater accuracy than the fabrication technology. Indeed, FIBs are intended to repair circuits, and must thus be able to alter them with a precision at least equal to that of the layout. This is normal because FIBs use a naively finer process: ions accelerated by FIBs have a smaller wavelength than the light used in lithographic fabrication processes. Thus, minimally-spaced M6 lines are employed for the chip protection.

Two other properties that an active shield must enjoy are:

1) it must cover the circuit uniformly, and
2) it must resist against alteration. From Fig. 5, it

is clear that the greatest the number the equipotentials, the more difficult "rerouting attacks" (that were sketched in Fig. 4). Also, altering the geometry without modifying the connectivity will be all the harder as the shield seems more entropic.

IV. Solution: Dense Random Spaghetti Active Shield

A. Rationale

The idea of dense random spaghetti active shield consists in:

- defining a set of vertices and edges, that correspond to a set of M6 and M5 sites that are encapsulated in the area over the module(s) to protect;
- such that whatever subgraph (provided the smallest non-convex components are not singletons) is DRC compliant.

Then, for the compactness property of the shield (absence of holes), it must be ensured that all the vertices are visited once (and at most once, so as to avoid short circuits), hence a Hamiltonian path. The complete algorithm is detailed in Alg. 1.

Algorithm 1 Dense Random Spaghetti Routing.

```
Input: $N$: number of different interleaved equipotentials.
Output: A random shield made up of $N$ equipotentials.
1: Build a graph whose vertices consist in free via slots and edges in the free routing slots.
2: Label each edge by a random number.
3: Solve the Traveling Salesman Problem (TSP) to get one Hamiltonian circuit.
4: Cut the Hamiltonian circuit into $N$ subpaths, and return those.
```

B. Comments on the Approach

At step 1, the graph typically uses the available topmost layers, like M5 and M6. At step 3, The TSP can be run until an exact solution is found, or it can be used to yield only an approximation. Indeed, our requirement is to end up with a Hamiltonian circuit, but not necessarily optimal in terms of length. At the end of step 3, the shield consists in a single equipotential, that can thus easily be cut and shunted at will by an attacker. Thus the need for step 4, that consists in segmenting the found Hamiltonian circuit into many (N) subpaths, that are interleaved, because the graph is randomly annotated.

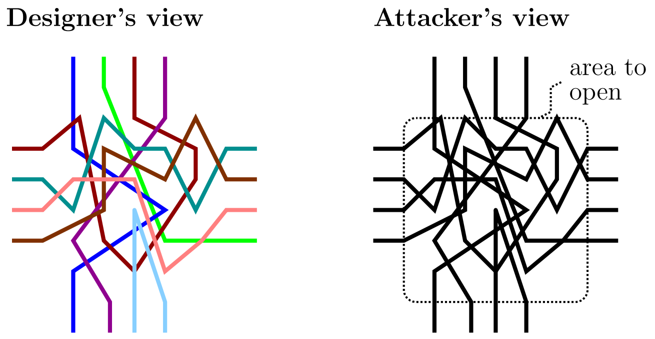


Figure 7. The figure on the *left* illustrates the N segments making up an active random shield. The connectivity of these segments (indicated with $N = 8$ different colors) is unknown to the attacker; the figure on the *right* shows the vision of an attacker who discovers the shield.

The number N of segments must be defined in accordance with the expected number of tries an attacker is willing to make in order to successfully edit the routing (of course without changing the connectivity!). Random shielding thus does not deter better than regular shielding (such as that displayed on the right side of Fig. 5) against an attacker that knows its layout. But random shielding is deterrent if its layout is unknown (for instance because its random routing makes it painful to unravel, as illustrated on Fig. 7). Indeed, an attacker needs to recognize all the N equipotentials as a preliminary phase to start with a rerouting attack. Said differently, random shielding makes the identification phase hard. But once the shield topology is exposed, its exploitation is easy. It can be argued that this protection belongs to "security by obscurity" techniques. It is indeed; but in the light of CC, it is valid: the identification phase is also quoted, and difficult identification improves the overall score of an evaluation. So, having admitted that the goal is merely to make the identification hard, we can assert that the spaghetti routing generated our Alg. 1 is efficient.

We focus in the sequel on special types of graphs, that we call cuboids. They consist in a juxtaposition of N_x, N_y, N_z cubes in the three dimensions. We consider the case where at least two dimensions exist (*i.e.* at least two sizes in N_x, N_y, N_z are greater or equal to two – otherwise no Hamiltonian circuit can exist because one vertex has degree one). For a cuboid to be Hamiltonian, $N_x \times N_y \times N_z$ must be even, that is to say that at least one amongst (N_x, N_y, N_z) must be even. Indeed, the value of $x + y + z$ and $x' + y' + z'$ of two adjacent vertices have different parity. Thus, any cycle must have a length of even parity; such cycle cannot pass through all the vertices of the graph if the total number of vertices ($N_x \times N_y \times N_z$ in our case) is odd. The reciprocal assertion is also true: if $N_x \times N_y \times N_z$ is even, then the graph is Hamiltonian. Without loss of generality, we assume that N_x is even. Then, Fig. 8(a) shows one solution in the case of $N_z = 1$. When $N_z > 1$, a Hamiltonian circuit can be derived from a $N_x, N_z \times N_y, 1$ cycle built as previously (see Fig. 8(b)), by folding the structure N_z times (see Fig. 8(c)). Obviously, the algorithm 1 will find better randomized Hamiltonian circuits.

Lastly, we underline that in Alg. 1, it is important that a Hamiltonian path is found first and cut afterwards. Proceeding the other way around, no guarantee on the existence of solutions would exist. For instance, a subcycle (that passes only through some vertices) could be found; but when removing it, the remain graph might be separated in two unconnected parts, which is not wanted security-wise. Indeed, it creates isolated areas of shield, while the goal is to try and spread the wires around the whole graph (not locally).

C. A Small Example

The result of an execution of Alg. 1 is shown in Fig. 9. The plot (a) illustrates the non-planar graph. It is made up of a 4×6 sites area for the topmost metal (say M6), represented as squares, and the equivalent 4 × 6 sites area in M5, represented as dots. In this example, the vias are possible everywhere. To summarize, this graph has 48 vertices (24 drawn as squares and as many drawn as dots) and 100 edges. In plot (b), one Hamiltonian path, obtained by step 3 of Alg. 1, is drawn in purple. Eventually, the final results is shown in plot (c). The previous Hamiltonian path is cut in $N = 3$ paths (drawn in red, green and blue) at positions indicated by arrows (always in the lower level, *i.e.* M5). Here, it can be seen that the three equipotentials are well interleaved.

The generated layout is shown in Fig. 10, where metal M6 is orange (resp. cyan), M5 yellow, and via V56 cyan (resp. gray) for Cadence Virtuosos

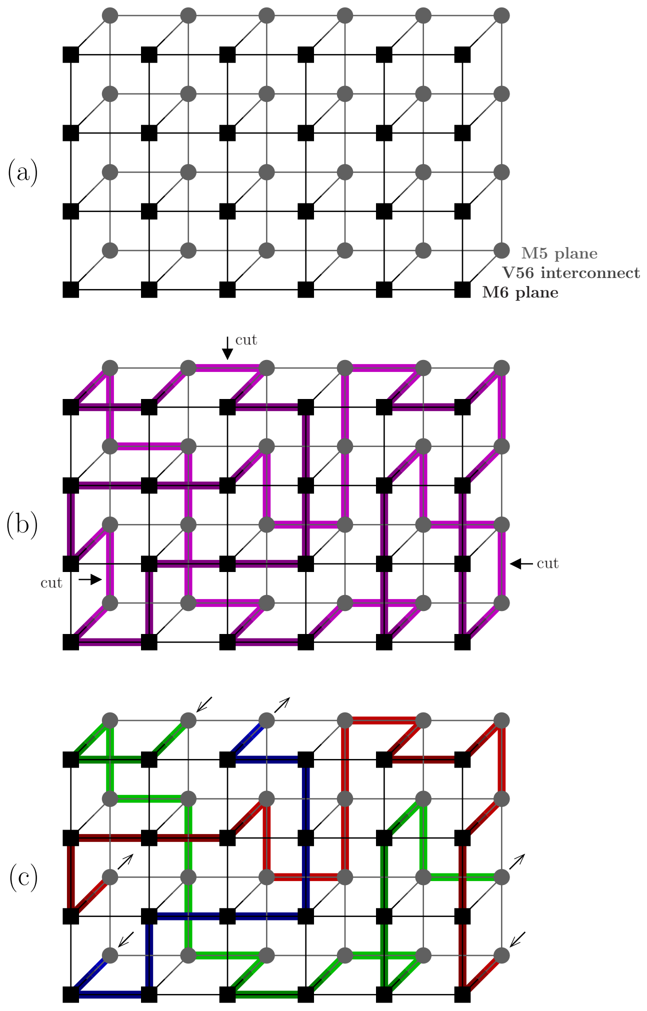


Figure 9. Compact shielding, obtained by the execution of Alg. 1. In the final shield layout (c), the $N = 3$ segments are fed with unrelated random bit sequences.

(resp. GNU/Electric). These layouts are drawn in HCMOS9GP 130 nm technology (STMicroelectronics). Given the DRC rules to be obeyed (see Sec. III-A), the shield drawing pitch is 97 nm, which is one order of magnitude smaller than the typical probe tip diameter ($\simeq 1\ \mu$m being the minimum achievable).

This kind of spaghetti routing fosters long lines, and is thus very difficult to unravel for 10+ different signals. Indeed, even if two equipotential wires are found, the overall cost to drill through a 10 × 10 pitch area requires nearly a complete shield reverse-engineering. Thus, we can assume this is not the primary attack path from an prospective attacker.

Incidentally, we notice that the shield lines can be

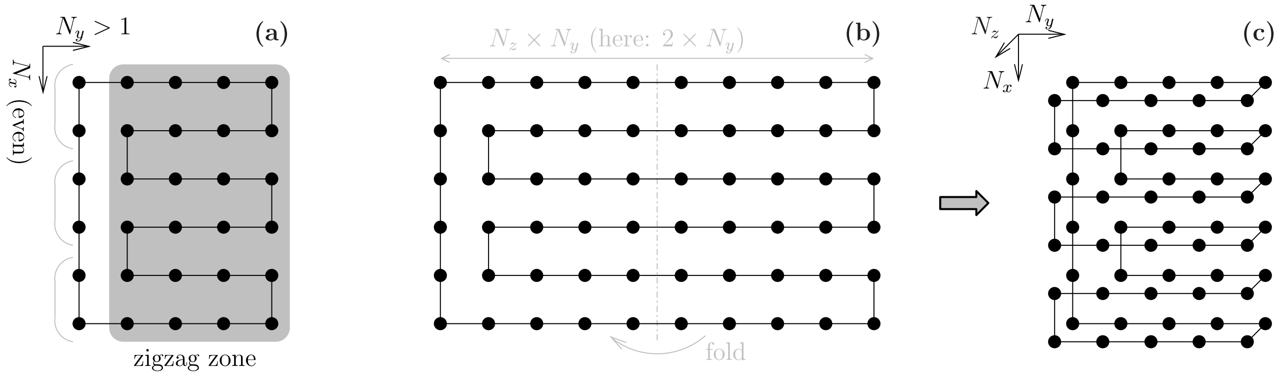


Figure 8. Constructive Hamiltonian paths when N_x is even.

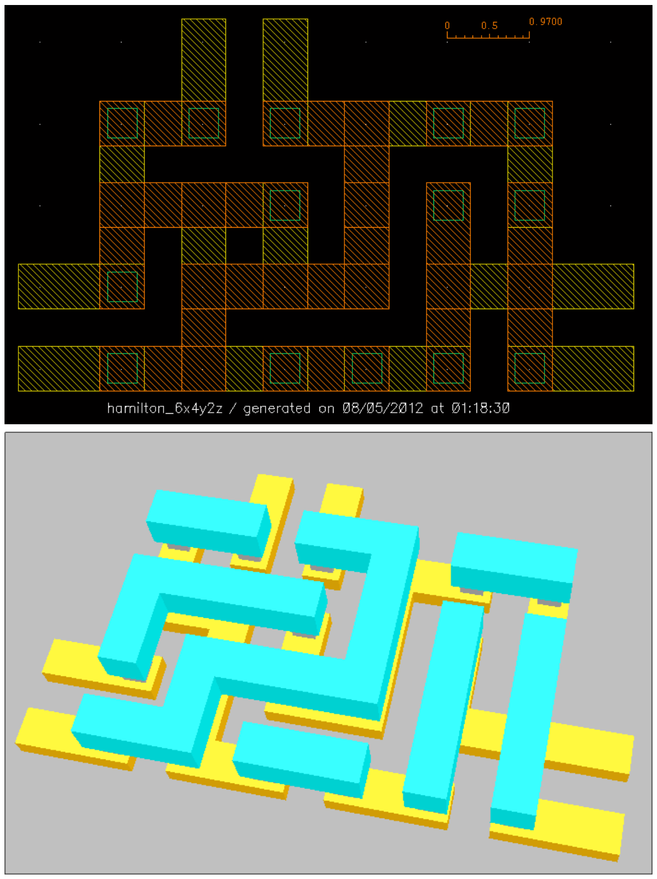


Figure 10. Shield design under the CADENCE VIRTUOSO and GNU/ELECTRIC layout editors.

heavily loaded. Indeed, at every via, that are massively instantiated (about for half of the connections in the shield), a resistance of about 1 Ω is added. However, the active shield does not require a speedy test. A functioning at 10 times the nominal frequency is still enough in most cases to ensure its usefulness. Additionally, as the capacitive load is high, this shield could also benefit from a passive integrity check. In this article, we leave this consideration as further research.

D. *Performance on Larger-Scale Circuits*

High performance of the shield circuit generation is of utmost importance. Namely, step 3 of Alg. 1 is known to be NP-complete. Nonetheless, we actually do not need the shortest Hamiltonian path, but only one. In this case, some heuristic algorithms exist. For instance, the LKH software (based on [5]) allows to find a Hamiltonian path, but the graphs cannot be weighted. Said differently, it solves the TSP where edges are weighted only by 0 or 1. This method is quite fast, and allows to find Hamiltonian circuits for up to several thousands vertices. Nonetheless, we observed that the result was not intricate enough.

Two key points for a scalable solution are thus to quantify the quality of a shield, and to find Hamiltonian circuits generation algorithms faster than those derived from the TSP.

1) Shield Quality: One feature to distinguish a random shield from a non-random one is the average isotropy of the routing. According to this criterion, the shield is all the better as the lines go almost equally in all directions. For this reason we propose the entropy of the directions as a metric. It is estimated as:

$$H(C) = \sum_{d\in\{x,y,z\}} -P(d) \cdot \log_2 P(d)\,, \qquad (1)$$

where $P(d)$ is the probability for the circuit to take this direction, evaluated as the ratio between the number of edges in that direction on the total number of edges (that is equal to the total number of vertices). When the shield is only 2D, $P(z) = 0$, thence we employ the limit $-P(z) \cdot \log_2 P(z) = \lim_{\epsilon\longrightarrow 0^+} -\epsilon \cdot \log_2 \epsilon = 0$. As along one direction, there are as many edges going forward and backward, we intentionally neglect the notion of orientation for d. The optimal values are 1.000 bit for a 2D shield and approximately 1.585 bit for a 3D shield. To the authors' best knowledge, the question whether those bounds are tight for Hamiltonian circuits is open. We also underline that for a finite shield, the Eqn. (1) is approximate, since vertices on the borders cannot have edges in all directions. However, in practical cases (see Tab. II), this metric is usable.

2) Genetic Algorithms: Genetic algorithms are random algorithms meant to approach an optimization problem by hybridizing solutions. In our case, they constitute a way to improve the Hamiltonian circuit's entropy. They are also interesting since we can start them from an existing Hamiltonian path (*e.g.* the one described in Fig. 8), and not from a hard-to-generate one obtained by exact or approximate TSP. Our mutation consists in randomly finding two pairs of adjacent lines, and to switch the connections, as illustrated in Fig. 11. This method requires a "bootstrap" Hamiltonian circuit (step 1). Then, it breaks the circuit into two circuits (step 2) twice. Eventually, the circuits are merged (step 3). The last step hopefully increases the Hamiltonian circuit's entropy. It is repeated until a sufficient entropy level is reached. For the sake of clarity, the same transformation is also shown in Fig. 12. The Hamiltonian cycle is represented in abstract way (there is no grid), and the pair of 4 neighbour sites are represented by squares (■) and by diamonds (◆): topologically speaking, it is possible to jump from one square (or diamond) to its neighbor with one hop. The original circuit uses the paths labelled ❶, ❷, ❸ & ❹ in this order (❶,❷,❸,❹). After transformation of the routing, the new circuit, (❶,❹,❸,❷), remains Hamiltonian. It requires that the squares and the diamonds be pairwise interleaved.

The table I shows that the genetic algorithm indeed manages to increase the entropy. It is illustrated on 10 iterations (from left to right, and from top to bottom), where it is also apparent that the entropy (*c.f.* Eqn. (1)) appears to be a suitable metric to measure the entanglement of the N segments making up the active shield.

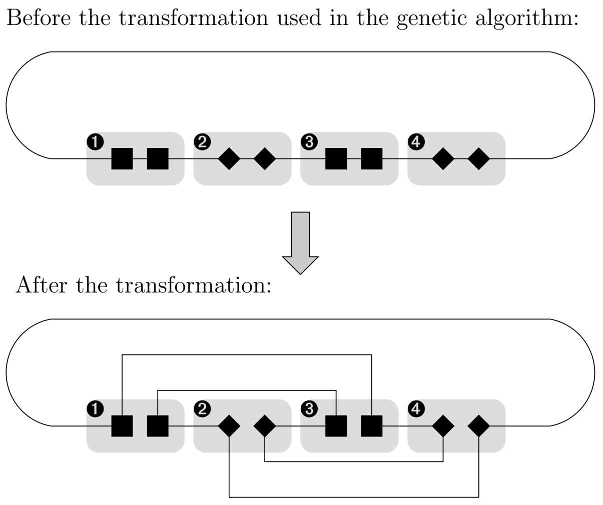

Before the transformation used in the genetic algorithm:
❶
❷
❸
❹
After the transformation:
❶
❷
❸
❹

Figure 12. Diversification process of Fig. 11, seen topologically.

The performance of the shield generation has been prototyped on three examples of interest for the smartcard industry:

1) a register containing a 128 bit key,
2) a 1 kB ROM, and
3) a DES cryptoprocessor.

In all cases, a $z = 2$ layer shield has been considered. The generation software is a prototype application, and thus written in a script language, without optimizations and all the "assertions checks" enabled. More precisely, it is a script PERL (version 5.10.1), executed on an Intel Xeon CPU cadenced at 2.13 GHz running GNU/Linux 2.6.32. The results are provided with in Tab. II. By rewriting the software with optimisations in mind, we could expect a two order of magnitude decrease in the execution time. The purpose of Tab. II is to show that with a limited memory footprint (less than 100 MB) and straightforward code, the active shield of large sizes can be generated by genetic algorithms. Eventually, Fig. 13 illustrates the convergence speed to a solution of highest entropy (or its vicinity). The progression sometimes stalls, as for instance for the DES shield in the early minutes. The reason is that when choosing a first transformation site (step 2 of Fig. 11), this selection can leave few choices for the

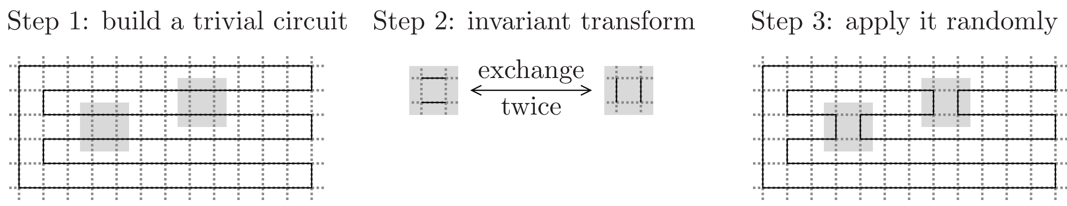

Step 1: build a trivial circuit
Step 2: invariant transform
exchange
twice
Step 3: apply it randomly

Figure 11. Diversification of Hamiltonian circuits; initial/final state is shown as (1)/(3), and the transformation (2) shall be applied twice or an even number of times.

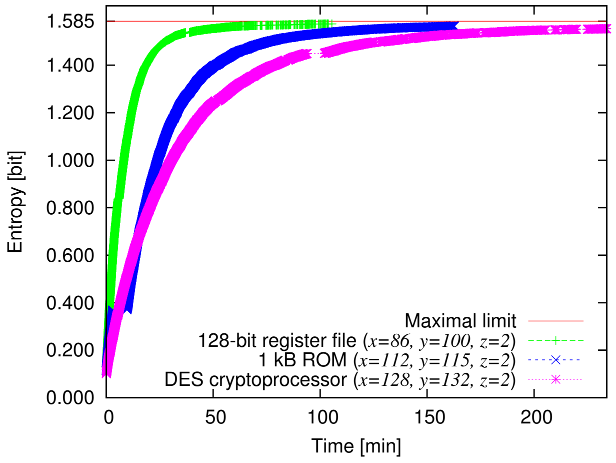


Figure 13. Convergence rate of three real-world random active shields.

second transformation site, hence numerous trials-and-errors. Also, it can be seen that the curves stop when a maximal value (for our version of genetic algorithm) is reached and no other positive mutation can be found. It is remarkable to notice that the final value of the entropy is very close to the maximal theoretic one, which proves the mutation method presented in Fig. 11 is relevant. We also notice that genetic algorithms allow for a trade-off *security* versus *computation time*. A less entropic shield can be found faster than an optimal (or near optimal) one, by stopping the genetic algorithm if the entropy is considered high enough (for instance, a threshold of 1.550 bit can be set).

Even larger circuits can be protected at a lower computational cost by abutting several instance of smaller active shield problem. Although less elegant, this solution still provides with an increased security level since the instance are *a priori* one from each other, making artificial FIB rerouting connections chancy.

V. Conclusions and Perspectives

The adequate shielding of secure integrated circuits is of great importance, in regard with current exploits, for instance from FlyLogic. Active shielding is a known technique, that consists in injecting random data through top-metal wires and checking that they arrive uncorrupted. However, most publicly disclosed active shields are structured, hence their topology can be inferred by attentive attackers. In this article, we propose a method to achieve intricate spaghetti routing of a dense mesh of wires. The density is actually optimal in the sense that no routing site is left empty. This is achieved by computing a Hamiltonian circuit. The entanglement comes from randomized constraints given in the Hamiltonian circuit generation algorithm.

It is worthwhile to notice that we have employed graph-agnostic algorithms (only the genetic one requires an initial Hamiltonian circuit). However, most of the time, the graphs are highly structured: for instance, they are often "lattices", *i.e.* the (finite) repetition of a fixed pattern. Certainly the algorithms can be guided to converge faster if they are aware of the graph's topology. But given that general (graph-agnostic) algorithms finish rapidly[2], this refinement is left as a perspective for future incremental improvement on top of Alg. 1.

Eventually, we notice that the active shield presented in this paper is statically random. An improvement could consist in changing the division of the Hamiltonian circuit into several N segments, depending on a "shield configuration" random variable. The feasibility of this dynamically random routing is of interest for the future generations of active shields.

Acknowledgments

The authors acknowledge the partial financial support of project **TOISE**, *Trusted Computing for Euro-*

[2]As compared with other CAD tools used in ASIC design.

Table II
COMPUTATION TIME TO GENERATE A HAMILTONIAN CIRCUIT THAT CAN SERVE AS SHIELD FOR SEVERAL SENSITIVE MODULES OF A SMARTCARD.

Circuit	**Area**	**Number of vertices**	**Time for the generation**	**Entropy — Eqn. (1)**
128-bit register file	$10,000\ \mu m^2$	$17,200$	1 h 45 min	1.574 bit
1 kB ROM	$15,000\ \mu m^2$	$25,760$	2 h 43 min	1.564 bit
DES crypto-accelerator	$21,000\ \mu m^2$	$33,792$	3 h 54 min	1.554 bit

pean Embedded Systems, funded under grant ENIAC-2010-1, and of project **MARSHAL+**, *Mechanism Against Reverse Engineering for Secure Hardware and Algorithm*, funded under grant FUI12.

REFERENCES

[1] Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408).
Website: http://www.commoncriteriaportal.org/.

[2] Andrea Beit-Grogger and Josef Riegebauer. Integrated circuit having an active shield, November 8, 2005. United States Patent number 6,962,294.

[3] Sébastien Briais, Stéphane Caron, Jean-Michel Cioranesco, Jean-Luc Danger, Sylvain Guilley, Jacques-Henri Jourdan, Arthur Milchior, David Naccache, and Thibault Porteboeuf. 3D Hardware Canaries. In *CHES*, Lecture Notes in Computer Science. Springer, September 9-12 2012. Leuven, Belgium. Full version in ePrint Archive, Report 2012/324 (http://eprint.iacr.org/2012/324/).

[4] Christophe Giraud. *Attaques de cryptosystèmes embarqués et contre-mesures associées*. PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, 26 octobre 2007. http://www.prism.uvsq.fr/fileadmin/CRYPTO/TheseCG-new.pdf.

[5] Keld Helsgaun. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[6] INVIA. Active Shield IP (*digital IP and analog IP that detects invasive attacks*). http://www.invia.fr/Active-Shield-23.html.

[7] Peter Laackmann and Hans Taddiken. Apparatus for protecting an integrated circuit formed in a substrate and method for protecting the circuit against reverse engineering, February 19 2003. United States Patent number 6,798,234.

[8] Martin Schobert. GNU software DEGATE. Webpage: http://www.degate.org/.

[9] Christopher Tarnovsky. How to Reverse-Engineer a Satellite TV Smart Card, 2010. Online video: http://www.youtube.com/watch?v=tnY7UVyaFiQ.

[10] Christopher Tarnovsky. Infineon / ST Mesh Comparison, February 14th 2010. http://www.flylogic.net/blog/?p=86.

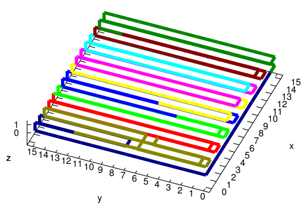
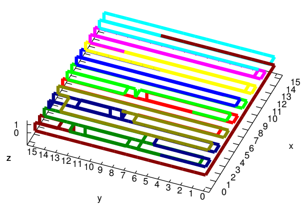
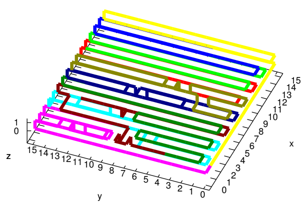
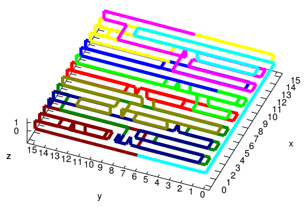
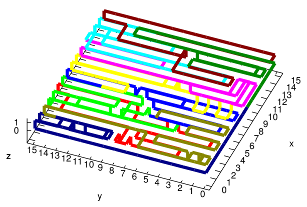
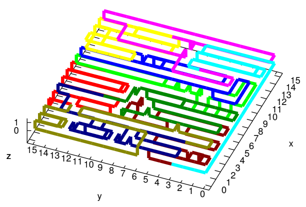
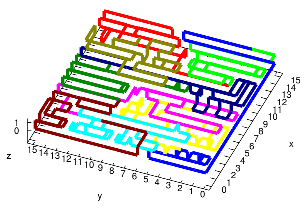
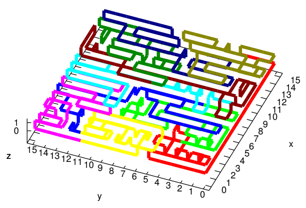
[11] Randy Torrance and Dick James. The State-of-the-Art in IC Reverse Engineering. In *CHES'2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 363–381. Springer.

[12] Pim Tuyls, Boris Skoric, and Tom Kevenaar. *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, December 2007. 1st Edition, ISBN 978-1-84628-983-5.

[13] Neil H.E. Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison Wesley, 2004. 3rd edition (May 11, 2004).

Table I

Evolution of a $x = 16$, $y = 16$, $z = 2$ shield with $N = 10$ segments (printed in different colors). Indicated are the entropy H and the time T for generation.

$H = 0.550$ bit, $T = 37$ ms.	$H = 0.673$ bit, $T = 129$ ms.
	
$H = 0.783$ bit, $T = 213$ ms.	$H = 0.903$ bit, $T = 316$ ms.
	
$H = 1.014$ bit, $T = 438$ ms.	$H = 1.126$ bit, $T = 599$ ms.
	
$H = 1.240$ bit, $T = 940$ ms.	$H = 1.349$ bit, $T = 1381$ ms.
	
$H = 1.454$ bit, $T = 2228$ ms.	$H = 1.556$ bit, $T = 4303$ ms.