Intellectual Property Protection for FPGA Designs with Soft Physical Hash Functions: First Experimental Results

Stéphanie Kerckhof*, François Durvaux*, François-Xavier Standaert*, Benoît Gérard*

* *ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium.*
stephanie.kerckhof;francois.durvaux;fstandae@uclouvain.be

***Abstract*—The use of Soft Physical Hash (SPH) functions has been recently introduced as a flexible and efficient way to detect Intellectual Property (IP) cores in microelectronic systems. Previous works have mainly investigated software IP to validate this approach. In this paper, we extend it towards the practically important case of FPGA designs. Based on experiments, we put forward that SPH functions-based detection is a promising and low-cost solution for preventing anti-counterfeiting, as it does not require any a-priori modification of the design flow. In particular, we illustrate its performances with stand-alone FPGA designs, re-synthetized FPGA designs, and in the context of parasitic IPs running in parallel.**

I. Introduction

Current electronic products are so complex that it is common for developers to purchase and use third party designs known as Intellectual Property (IP) cores. A study from Semico Research estimated that the IP core market reached 4 billion US$ in 2009, i.e. a 23.2% growth compared to the 2005 figures [15]. The business model usually combines access to technology fees with royalties per sold units. It naturally raises important questions regarding the protection of these designs against unauthorized exploitation. Various solutions have been introduced to mitigate such risks. But as usual in security-related issues, the protection mechanisms have to deal with contradictory goals. On the one hand, IP owners primarily aim to prevent the piracy of their designs. On the other hand, users want to easily integrate these designs in their development flows. The first goal suggests integrating security mechanisms at low abstraction levels (e.g. in circuit layouts). The second goal rather suggests integrating these mechanisms at high abstraction levels (e.g. in the description codes of the designs). In other words, solutions to prevent the counterfeiting of electronic systems can be seen as a trade-off between security and flexibility.

Examples of such solutions include permission-based and watermarking-based techniques. In the first case, the system performs some tests before running, in order to make sure that it has the right permissions. Passwords can be used for this purpose. Checking the presence of a cryptographically-enhanced security chip, or exploiting Physically Unclonable Functions (PUFs), are harder to bypass alternatives [3], [7], [13], [16]. In the second case, a piece of information (called the watermark) is embedded in the IP, in order to verify its authenticity or the identity of its owners. Watermarks have to be both robust (i.e. detectable even if the IP is slightly modified) and imperceptible (i.e. the protected design should have the same functional behavior and similar performances as the original one) [2], [8], [9], [12]. In this context, a recent proposal has been to exploit physical features of the IPs (such as their power consumption) to detect the watermarks [4], [18]. In contrast with permission checks which prevent the run of illegal designs, watermarks are used *a posteriori*, in order to detect fraudulent copies. One limitation of these approaches is that they require the inclusion of a security mecanism within the IP to protect, hence to modify the original layout. This implies performance overheads and can hardly prevent counterfeiting by adversaries who can manipulate the source code (if they can remove the security part of the designs). Also in terms of flexibility, permissions checks and watermarks are not optimal since they require to include the IP-protection early during the design process.

More recently, an alternative proposal has been introduced, taking advantage of Soft Physical Hash (SPH) functions [5]. Similarly to [4], [18], it exploits physical features of the IP to protect. But contrary to these watermarking-based approaches, it aims to detect counterfeited designs without any addition to the original layout. The central idea of SPH functions (that is also borrowed from the field of image processing [6], [11], [14], [17]), is that the designs themselves should be specific enough for being their own “signature", and that this specificity can be detected from physical measurements. The main features required for SPH functions-based anti-counterfeiting to be effective are the *content sensitivity* and the *perceptual robustness* of the hashes. That is, SPH should in the same time be robust against an (owner-defined) set of IP-preserving transformations (such as changing the variables names or compiler options in a software implementation), and sensitive to more significant variations of the IP (e.g. significantly modifying its architecture or functionality). In [5], the idea of SPH function-based anti-counterfeiting has been validated with a software case-study. Namely, the authors showed that the IP of 10 different block cipher implementations in an Atmel microcontroller could be detected from their power consumption, with sufficient content sensitivity and perceptual robustness (e.g. against simple code transformations such as changes of registers, instruction swapping or dummy oper-

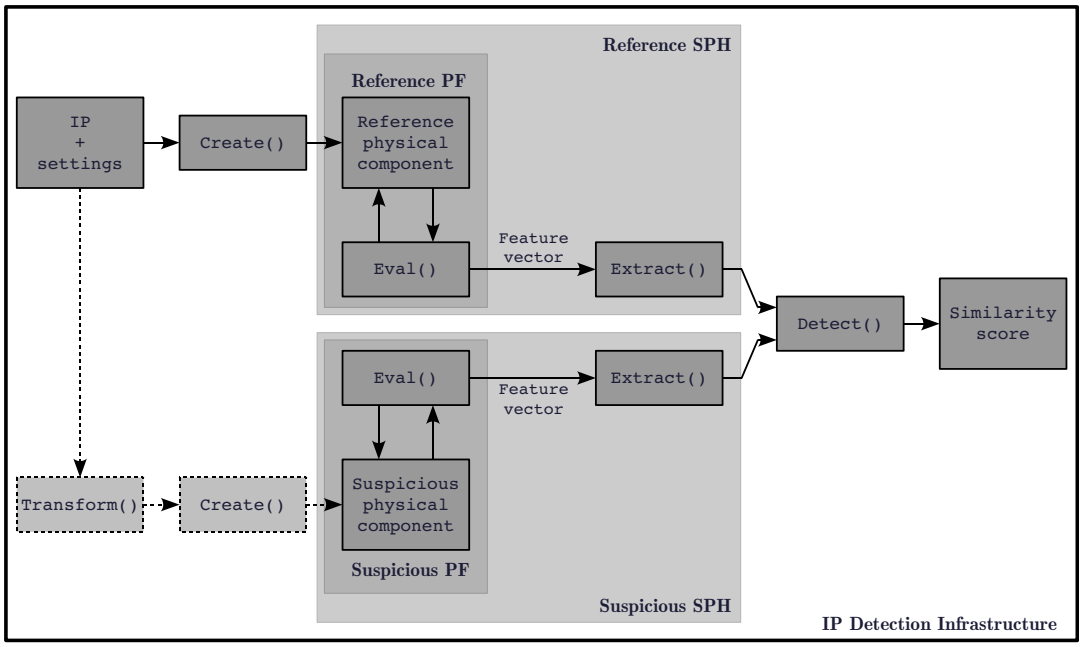

Figure 1. Generic framework for IP detection.

ations). Quite naturally, the extension towards a hardware context was one of the main open challenges, since the IPs generally have higher value in this case, because of longer development times and more complex optimizations.

In this paper, we consequently extend the evaluation of SPH function-based anti-counterfeiting to the meaningful case of FPGA implementations. For this purpose, we considered five lightweight block ciphers (namely HIGHT, ICEBERG, KATAN, NOEKEON and PRESENT) together with the standard AES Rijndael, of which the ASIC designs were presented at CHES 2012 [10]. All these algorithms were implemented according to a similar architecture aiming at energy efficiency, implementing one to a couple of rounds per clock cycle depending on the ciphers. This case-study was mainly motivated by the goal of analyzing a challenging set of potentially similar IP cores. We additionally considered different realistic scenarios such as stand-alone designs, re-synthetized designs and designs with parasitic IPs running in parallel. Experimental results suggest that the SPH approach for IP detection remains a promising and flexible solution for FPGA implementations. Depending on the cases, we successfully detected the IPs without any knowledge of their inputs, or taking advantage of the data dependencies in the power traces generated by the running algorithms, with limited number of measurements.

II. Background

The framework for SPH function-based IP detection is represented in Figure 1. Its specification starts by choosing an object to protect that can be any type of IP (e.g. a source code, a netlist, a layout, ...). This object to protect is then embedded into an implementation (the next sections will consider an FPGA case-study) and the combination of this implementation with an evaluation process (i.e. a measurement setup) constitutes what we denote as a physical function (PF). The physical function outputs a feature vector which is any physical emanation of the target device running the IP. From this feature vector, we use an extraction procedure, i.e. a signal processing tool of which the goal is to produce an output that best represents the IP. The output of the extraction procedure is called the SPH (hash for short).

As suggested by the figure, the detection of counterfeited IPs essentially works by comparing different hash values. That is, we assume that the IP owner has characterized the SPH function of its design. Note that this characterization does not have to be done during development time, as the decision to use SPH can be taken even after a product has been released. Then, given any suspicious IP, a detection procedure will be used to output a level of similarity between the two hash values. As indicated by the dotted parts of the figure, the suspicious IPs may directly correspond to the counterfeited designs, or to slightly transformed ones. Eventually, the performances of the detection framework are measured with the previously mentioned content sensitivity and perceptual robustness, for which definitions are given in [5]. Informally, the central element of these definitions is the need to specify a set of IP-preserving transformations. Typically, some trivial modifications of the code should not be considered as leading to a new IP. In the following, we will consider the re-synthesis of a design under a different set of constraints, and the addition of a parasitic IP running in parallel, as IP-preserving transformations. By contrast,

a change of block cipher is naturally considered as non-IP-preserving. Given this choice, the perceptual robustness is the probability that two designs that only differ by IP-preserving transformations lead to a similarity score larger than a certain threshold τ, and the content sensitivity is the probability that two designs that differ by non-IP-preserving transformations lead to a similarity score lower than a certain threshold τ'. These thresholds do not have to be identical, but the perceptual robustness one has to be higher than the content sensitivity one for the detection to succeed.

Such a generic detection framework can be run in different contexts, more or less favorable to the IP owner. For example, the inputs of the design to protect and its source code can be known or unknown during detection, and the framework can be applied to identical or different technologies, using identical or different measurement setups.

III. FPGA CASE-STUDIES

In this section, we analyze the application of our SPH function-based IP detection framework to FPGA designs. As previously mentioned, we considered three main scenarios for this purpose. First, we focused on the simple(st) case where the suspicious IPs are stand-alone FPGA designs corresponding to the object to protect. Second, we studied the case where the counterfeiter has re-synthetised the object to protect under a different set of constraints. Finally, we evaluated a more challenging (and realistic) context where the illegal IP is integrated in a larger system, with a parasitic design running in parallel. In the three scenarios, we first specify the components of the IP detection infrastructure we used, and then present experiments confirming its interest.

A. Stand-alone FPGA designs

1. Specification of the IP detection infrastructure. We used the bitstreams of five lightweight ciphers (HIGHT, ICEBERG, KATAN, NOEKEON and PRESENT) and the AES Rijndael as *objects to protect*. These bitstreams were obtained from the unrolled architectures described in [10], using the number of rounds per cycles that leads to maximum energy efficiency for each cipher, and synthetized for a Xilinx Virtex-II Pro FPGA. In this subsection, we used a default set of place-and-route options, leading to the performance results indicated in the left column of Table I. We additionally used the FPGA power consumption as *physical feature vector*. Measurement traces were obtained by measuring the voltage variations around a shunt resistor provided on the Sasebo-G board [1]. The device was running at a frequency of 24 MHz, and the oscilloscope sampling frequency was set at 2,5 GHz. In the following, we will characterize the SPH of each reference IP core with a feature vector made of the power consumption corresponding to one complete encryption. As for the suspicious IPs, the traces were simply obtained by measuring repeated encryptions. In this first case of stand-alone FPGA designs, the *extraction* phase directly outputs the complete traces (without performing any signal processing), and the *detection* process uses Pearson's correlation coefficient. That is, let x and y be two measurement vectors of length n with mean values $\bar{x}$ and $\bar{y}$, this coefficient is defined as:

$$\rho(x,y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})\cdot(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}.$$

As reference traces and suspicious ones do not always have the same length, we further cropped the suspicious traces whenever longer than the reference ones[1]. The *context* in which the IP detection infrastructure is used is the following: the inputs provided to the IPs are unknown, we do not have access to the source code and the same device and measurements setup were used for all the tests.

Table I
IMPLEMENTATION RESULTS FOR THE SIX CONSIDERED IPS WITH TWO SETS OF SYNTHESIS OPTIONS (DEFAULT AND AREA CONSTRAINED).

	Default		Constrained	
	slices	T [ns]	slices	T [ns]
AES	1843	10.48	1687	9.28
HIGHT	573	7.76	596	8.13
ICEBERG	1123	9.04	1022	7.95
KATAN	725	7.02	650	6.87
NOEKEON	1125	5.94	1255	6.7
PRESENT	561	5.09	541	4.5

2. Experimental Results. Our first results are illustrated by Figure 2, where each column is linked to one particular suspisious IP compared to the six possible reference IPs. Each point in the figure corresponds to one experiment, i.e.

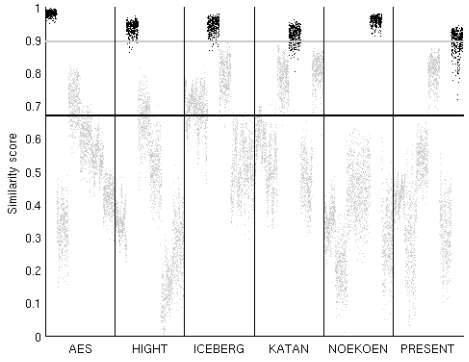

Figure 2. Similarity score scatter plot for stand-alone designs. Single reference traces / single suspicious traces / unknown inputs.

one similarity score obtained from the comparison between one reference trace and one suspicious trace. For each suspicious-reference IP pair, 400 experiments have been performed, represented by black dots whenever considering identical IPs, and grey dots otherwise. We additionally

[1]This assumes the traces to be synchronized, which can be achieved by different means, e.g. computing the correlation over a sliding window.

plotted the content sensitivity threshold (i.e. the maximum score for different IPs) with a grey line, and the perceptual robustness threshold (i.e. the minimum score for similar IPs) with a black line. As can be observed, the performances in this context are already reasonably good. Yet, some non-detections and false alarms occur for “close IPs", assumably because of measurement noise in our traces. The closeness between different IPs is further emphasized in Figure 3, where each square in the matrix corresponds to the mean similarity score between the IPs, and the higher similarity scores are indicated with a darker color code. Eventually,

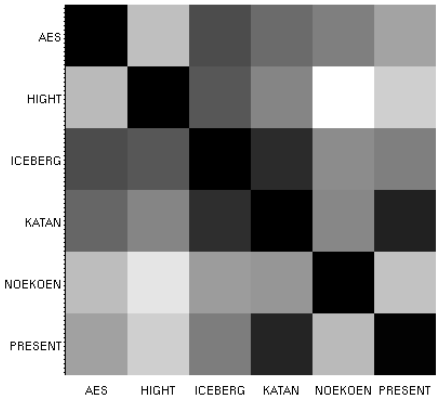

Figure 3. Similarity score correlation matrix for stand-alone designs. Single reference traces / single suspicious traces / unknown inputs.

in order to improve our performances and get rid of the noise, a natural solution is to exploit averaged traces rather than single traces, both for the reference and suspicious IPs. The result of this additional experiment (with a 10 times averaging) is given in Figure 4. In this latter case, we clearly see that no false alarm nor non-detections happens anymore, and the threshold for perceptual robustness is now higher than the one for content sensitivity in all cases.

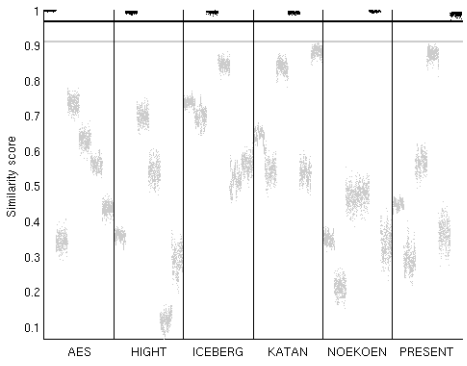

Figure 4. Similarity score scatter plot for stand-alone designs. 10 times averaged reference traces and suspicious traces / unknown inputs.

B. Re-synthetized FPGA designs

1. Specification of the IP detection infrastructure. The results in the previous subsection were considering similar algorithms for which the IP cores could be detected even in a simple context (unknown inputs and source code). Quite naturally, an important question is whether such an observation is still valid in the more challenging scenario where a potential counterfeiter would apply some IP-preserving transformations. This is the quesion we tackle in this subsection, considering the placement and routing of our designs with a different set of constraints as an IP-preserving transformation. Note that in this case, the object to protect is not the bitstream anymore, but rather the source code or netlist. For this purpose, we used both sets of constraints in Table I. Since the previous IP detection infrastructure turned out to be insufficient in this case, we additionally considered two tweaks to improve detection. First, we moved to a known input context, allowing us to take advantage of data dependencies in the traces. Next, we used a more sophisticated extraction procedure, based on a selection of Points Of Interest (POIs). Namely, we split the traces into consecutive clock cycles, using the Fast Fourier Transform to recover the rising edges of the clock signal. This was achieved by filtering the frequency spectrum around the clock frequency and its harmonics, then applying the inverse transform on the filtered signal. This preprocessing provides a sequence of peaks indicating where the rising edges of the clock signal are. Following, we were able to work on a sequence of clock cycles instead of raw side-channel traces. From this sequence, we finally extracted POIs having maximum Signal-to-Noise Ratio (SNR) for each cycle (with the SNR defined as the variance over mean traces for different plaintexts divided by the noise variance).

2. Experimental Results. For illustration, we first provide the similarity score correlation matrix of our re-synthetized designs obtained from the IP detection infrastructure of the previous subsection in Figure 5. Each line/column is now divided in two, for the two sets of synthesis options. As expected, the re-synthesis is not perfectly IP-preserving in this case. For example, the re-synthetized PRESENT design appears as a new IP. Interestingly, this issue is not really caused by differences in performances, but rather by the placement-and-routing illustrated in Figure 6. By contrast, taking advantage of the known input context with the selection of POIs proposed in the previous paragraph directly allows getting rid of this limitation. This is illustrated in Figure 7 in which we only analysed the IPs giving low similarity scores in the first detection scenario. A natural explanation for the obtained results is that the operation dependencies in the power consumption, that were exploited in the previous section, are significantly affected by re-synthesis. By contrast, data dependencies that are additionally considered in the known input scenario are more

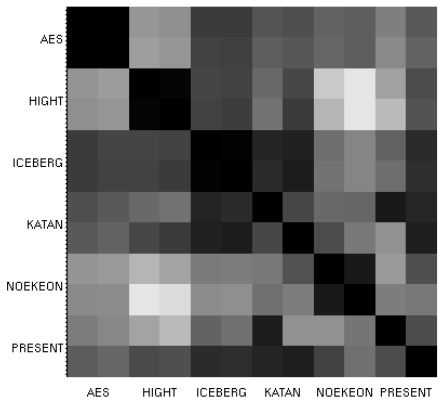

Figure 5. Similarity score correlation matrix for re-synthetized designs. 10 times averaged reference traces and suspicious traces / unknown inputs.

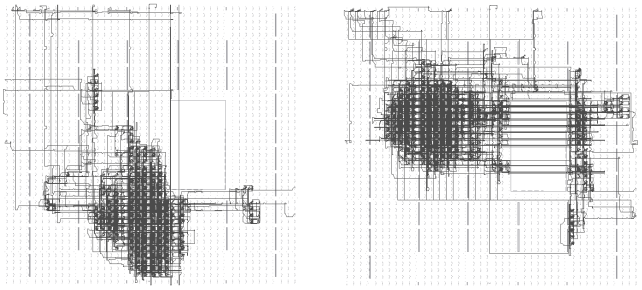

Figure 6. Floorplan of the PRESENT designs.

robust against such transforms (since the same data has to be manipulated). In this respect, an even more challenging type of transforms would try to affect the intermediate data during the computations of an IP, without affecting its final result. We leave their investigation for further research. Note that the exploitation of data dependencies crucially depends on a good selection of POIs. For illustration we additionally provided the weaker detection results obtained with an uninformed selection in Appendix, Figure 9.

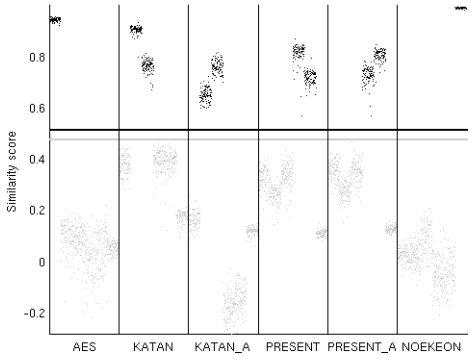

Figure 7. Similarity score scatter plot for re-synthetized designs. Single reference traces / single suspicious traces / known inputs.

C. Parasitic IP running in parallel

1. Specification of the IP detection infrastructure. Before concluding this work, we finally considered the practically important case study, where not only our suspicious IP would run on an FPGA, but also some parasitic one. As a preliminary investigation in this direction, we investigated the case of a Linear Feedback Shift Register (LFSR) running in parallel to PRESENT (we limited our evaluations to this IP because it was the most challenging one in the previous subsection). The performances of the different suspicious IPs with such an additional LFSR are given in Table II for three different LFSR sizes. For the rest, we used the same IP

Table II
IMPLEMENTATION RESULTS FOR PRESENT WITH PARALLEL LFSR.

	Default	
	slices	T [ns]
PRESENT + 64-bit LFSR	608	4.87
PRESENT + 512-bit LFSR	797	4.93
PRESENT + 1024-bit LFSR	1053	4.76

detection infrastructure as in the previous subsection. The only difference is that we will need to work with (5 times) averaged traces in order to get rid of the “algorithmic noise" coming from the parallel execution of the LFSRs.

2. Experimental Results. The scatter plot of this final experiment is given in Figure 8, where we used the same subset of reference IPs as in Figure 7. In all three cases and despite the presence of a parasitic LFSR, we were able to detect the suspicious PRESENT. Interestingly, we can also observe the impact of larger algorithmic noise, as the distance between the content sensitivity and perceptual robustness thresholds decreases with the LFSR size. As a counterfeited IP will be integrated in even larger designs, we can expect that the detection will still succeed given that larger amounts of traces are averaged during extraction.

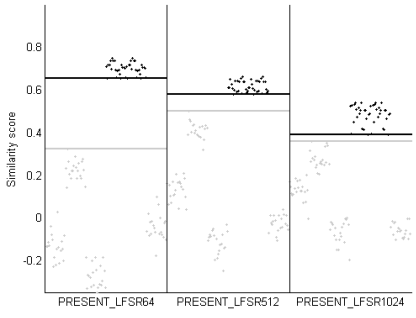

Figure 8. Similarity score scatter plot for designs with parasitic IP. 5 times averaged reference traces and suspicious traces / known inputs.

IV. Conclusions

We believe that the main interest of the IP detection mechanism discussed in this work is its flexibility. First, it does not require to modify any piece of the original designs. Second, the decision to test a suspicious IP can be taken a posteriori (without beging considered at development time). Eventually, it is potentially useful against adversaries that can operate on high-level IP (e.g. source codes or netlists). By contrast, permission-based solutions and watermarks are hardly effective in this case. Quite naturally, this flexibility comes at the cost of sometimes difficult detection. While preliminary experiments in this work are promising, considering more challenging scenarios (e.g. different technologies, other parasitic IPs, other measurement setups) is certainly important. It is likely that advanced extraction and detection tools will be needed in these cases. Yet, we hope that SPH functions-based IP detection can be a useful element to solve the challenge of counterfeited electronic designs, combined with other ideas with different strenghts and weaknesses.

Acknowledgements. This work has been funded by the Walloon region WIST program project MIPSs. François-Xavier Standaert is an Associate Researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). Stéphanie Kerckhof is a PhD student funded by a FRIA grant, Belgium.

References

[1] http://staff.aist.go.jp/akashi.satoh/sasebo/en/index.html.

[2] Amr T. Abdel-Hamid, Sofiène Tahar, and El Mostapha Aboulhamid. A survey on IP watermarking techniques. *Design Autom. for Emb. Sys.*, 9(3):211–227, 2004.

[3] Catalin Baetoniu. FPGA IFF copy protection using Dallas semiconductor/Maxim DS2432 secure EEPROMs. XAPP780, May 28, 2010.

[4] Georg T. Becker, Markus Kasper, Amir Moradi, and Christof Paar. Side-channel based watermarks for integrated circuits. In Jim Plusquellic and Ken Mai, editors, *HOST*, pages 30–35. IEEE Computer Society, 2010.

[5] François Durvaux, Benoît Gérard, Stéphanie Kerckhof, François Koeune, and François-Xavier Standaert. Intellectual property protection for integrated systems using soft physical hash functions. In the proceedings of WISA 2012, Lecture Notes in Computer Science, vol xxxx, pp yyy-zzz.

[6] Jiri Fridrich and Miroslav Goljan. Robust hash functions for digital watermarking. In *ITCC*, pages 178–183. IEEE Computer Society, 2000.

[7] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA intrinsic PUFs and their use for IP protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.

[8] Andrew B. Kahng, John Lach, William H. Mangione-Smith, Stefanus Mantik, Igor L. Markov, Miodrag Potkonjak, Paul Tucker, Huijuan Wang, and Gregory Wolfe. Watermarking techniques for intellectual property protection. In *DAC*, pages 776–781, 1998.

[9] Andrew B. Kahng, Stefanus Mantik, Igor L. Markov, Miodrag Potkonjak, Paul Tucker, Huijuan Wang, and Gregory Wolfe. Robust IP watermarking methodologies for physical design. In *DAC*, pages 782–787, 1998.

[10] Stéphanie Kerckhof, François Durvaux, Cédric Hocquet, David Bol, and François-Xavier Standaert. Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 390–407. Springer, 2012.

[11] Frédéric Lefèbvre, Jacek Czyz, and Benoit M. Macq. A robust soft hash algorithm for digital image signature. In *ICIP (2)*, pages 495–498, 2003.

[12] Matthew Lewandowski, Richard Meana, Matthew Morrison, and Srinivas Katkoori. A novel method for watermarking sequential circuits. In *HOST*, pages 21–24. IEEE, 2012.

[13] Bernhard Linke. Xilinx FPGA IFF copy protection with 1-wire® SHA-1 secure memories. XAPP3826, July 21, 2006.

[14] Vishal Monga and Brian L. Evans. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. *IEEE Trans. on Image Processing*, 15(11):3452–3465, 2006.

[15] Semico Research. Semiconductor intellectual property: The market hits its stride. http://www.design-reuse.com/news/11069/semico-research-report-semiconductor-intellectual-property -market-hits-stride.html, retrieved on November 20, 2012.

[16] Eric Simpson and Patrick Schaumont. Offline hardware/software authentication for reconfigurable platforms. In Louis Goubin and Mitsuru Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 2006.

[17] Ramarathnam Venkatesan, S.-M. Koon, Mariusz H. Jakubowski, and Pierre Moulin. Robust image hashing. In *ICIP*, 2000.

[18] Daniel Ziener and Jürgen Teich. Power signature watermarking of IP cores for FPGAs. *Signal Processing Systems*, 51(1):123–136, 2008.

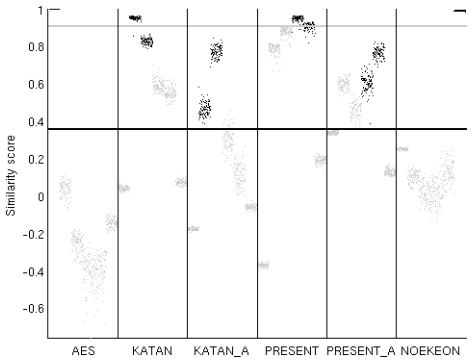

Figure 9. Same as Figure 7 with different POIs.