Overlay-Based Side-Channel Countermeasures: A Case Study on Correlated Noise Generation

Austin Baylis, Greg Stitt, Ann Gordon-Ross
Department of Electrical and Computer Engineering
NSF Center for High-Performance Reconfigurable Computing (CHREC)
University of Florida, Gainesville, FL, USA

***Abstract*—Side-channel attacks against field-programmable gate arrays (FPGAs) enable attackers to reverse-engineer bitfile encryption keys, resulting in intellectual-property (IP) theft and tampering. To address this problem, we demonstrate that overlays—virtual architectures implemented atop an FPGA—provide a novel countermeasure strategy that can protect application IP even on vulnerable FPGAs. Although we demonstrate such protection via a case study on correlated noise generation, the approach is potentially applicable to any countermeasure and any overlay. By extending existing overlay benefits (e.g., fast compilation, application portability, 1000x smaller bitfiles) with improved hardware security, our approach provides an attractive platform for Internet of Things, defense, and many embedded applications.**

Keywords—overlay, FPGA, differential power analysis, hardware security, virtualization, side-channel attacks

I. Introduction

Field-programmable gate arrays (FPGAs) are increasingly used for embedded and data-center applications due to their performance, power, and/or energy advantages over other technologies [3]. However, FPGAs suffer from hardware-security concerns that enable attackers to perform side-channel attacks that can reverse-engineer encryption keys, extract intellectual property (IP), and maliciously tamper with functionality. Such attacks are potentially devastating for Internet of Things (IoT) applications, where billions of deployed units can be maliciously modified. Similarly, attacks on safety-critical applications in defense and automotive applications can result in loss of human life.

One of the main limitations of existing countermeasures is a lack of protection for the FPGA's encryption key that is used to decrypt application bitfiles during device configuration. Such protection is challenging because FPGAs provide no mechanisms to control functionality during configuration, where the device is in an undefined state. Although there are countermeasures provided by AES cores (e.g., [9]), existing FPGAs are not manufactured with these cores, making existing FPGA bitfiles vulnerable to reverse engineering. Some FPGA vendors have introduced specialized FPGAs with various countermeasures [12], but such devices are still vulnerable to attacks [12], while also having cost disadvantages compared to commercial-off-the-shelf (COTS) FPGAs.

To address this concern, we present an approach that overcomes the lack of protection for the FPGA's key (or any secret information) by using a virtual architecture implemented on the FPGA (i.e., an *overlay*). Overlays have been introduced for a variety of purposes including fast placement and routing, application portability, simplified debugging, among others [2][4][5]. A recent hardware-security study investigated using unique overlays to limit damage from tampering [13]. In this paper, we complement earlier overlay studies by demonstrating that integrating side-channel attack countermeasures into overlays can protect application IP on potentially any COTS FPGA.

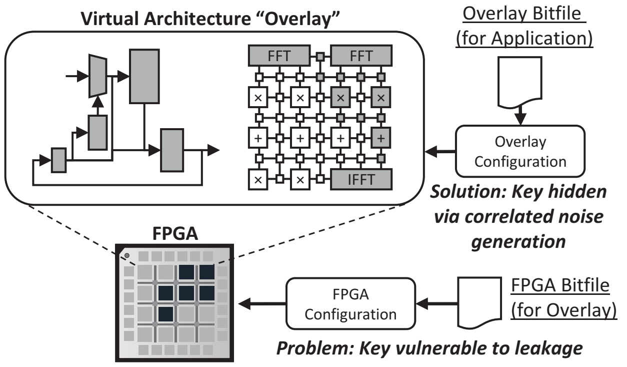
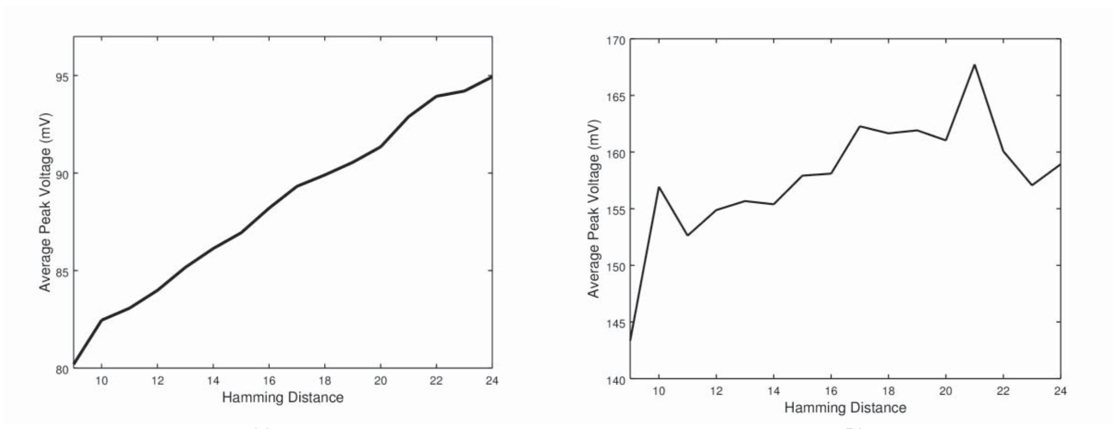

Virtual Architecture "Overlay"
FFT
FFT
IFFT
Overlay Bitfile (for Application)
Overlay Configuration
Solution: Key hidden via correlated noise generation
FPGA
FPGA Configuration
FPGA Bitfile (for Overlay)
Problem: Key vulnerable to leakage

Figure 1: Overview of overlay-based correlated noise generation. In our approach, the FPGA provides an application-specialized virtual architecture (i.e., an *overlay*). Whereas FPGA configuration is vulnerable to power-analysis attacks, overlay configuration protects encryption keys by injecting input-correlated power noise into the FPGA.

Figure 1 provides an overview of our approach, which implements applications on potentially any overlay as opposed to directly on the FPGA. Because of this two-layer approach, the system uses two separate bitfiles: 1) a vulnerable FPGA bitfile that configures the FPGA with the overlay, and 2) the protected overlay bitfile that contains all application IP. Although the FPGA's encryption key is still vulnerable, if an attacker were to discover the FPGA key, reverse engineering the bitfile would only provide the architecture of the overlay, which provides little application information [13]. The main security advantage of overlays comes from the fact that, unlike FPGA configuration, overlay configuration has control over the functionality of the FPGA, which we can exploit to provide various countermeasures.

To demonstrate overlay security advantages, we perform a case study on statistical power-analysis attacks. Such attacks (e.g., differential [10] and correlation power analysis [1] attacks) allow attackers to reverse-engineer encryption keys via

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC- 0642422 and IIP-1161022.



(a) **(b)**

Figure 2: Correlation power analysis relies on (a) a relationship between Hamming distance and voltage, which we (b) obfuscate using correlated noise generation to prevent leakage.

statistical analyses of power consumption during operations that access a key (e.g., [8][9]). Although previous work has introduced numerous countermeasures [11], statistical power analysis is still a common concern, even for FPGAs marketed as secure [12].

In the presented case study, we control the functionality of the FPGA during overlay configuration by injecting power via toggling of nets. By correlating this power with an input to be encrypted/decrypted (i.e., *correlated noise generation*), our approach provides the same noise for repeated attacks using the same input, and different noise for different inputs, which complicates differential and correlation power analysis by yielding correlations with an incorrect key. We evaluate the countermeasure against correlation power analysis attacks on DES using publicly available power traces [6] and show that such attacks yield incorrect keys.

Although we demonstrate the approach for noise generation, which is known to have limited effectiveness as a countermeasure [7], the same strategy can potentially be used for any existing countermeasure, which we will evaluate in future work.

II. Statistical Power Analysis Countermeasure

This section presents the details of statistical power analysis (Section II.A) and our countermeasure that can be integrated into overlays. The approach consists of two parts: 1) correlated noise generation to determine the type of noise to be injected to hide information leakage (Section II.B), and 2) a power injector core that can increase and decrease power by specific amounts each cycle (Section II.C).

A. Statisical Power Analysis

Statistical power analysis attacks create a statistical relationship between changes that occur during an operation and the measured power to extract leaked information. The classical statistic power analysis is differential power analysis (DPA) [10], where the attacker guesses input values for a model of the operation they wish to attack to generate potential output values. One bit of the output value is used to sort the power traces into two sets. The attacker then takes the difference of the mean of each set. Incorrect guesses average to be similar, resulting in a small difference, while a correct guess results in a large peak at the location the operation occurs.

An improvement on this attack is to approximate power consumption using the Hamming weight of all the bits of the output value generated by the input guess. The Hamming weight can then be used to build a linear correlation between the guess and the captured power traces. The correct guess would be associated with the highest correlation. Both previously mentioned methods of DPA, however, are limited due to unrealistic modeling of power.

Therefore, the attack we used to test the effectiveness of our correlated noise generator is correlation power analysis (CPA), as described in [1]. Instead of using Hamming weight, CPA uses Hamming distance because data leakage depends on the number of bits that flip from one state to another. CPA utilizes a model that assumes a bit flipping from 1 to 0 requires the same amount of energy as a bit flipping from 0 to 1. It is worth noting that while we explore CPA in this paper, CPA countermeasures tend to provide similar defensive efficiency against the other statistical analysis attacks [1].

Regardless of the method used, statistical power analysis attacks rely on the linear relationship between power consumption and the Hamming distance between the reference state and the variable output of the model. This relationship is demonstrated in Figure 2(a), which shows the average peak voltage during round one of DES versus the Hamming distance between the 32-bit right half at the start of the round and the 32-bit right half after the round. It is this linear relationship of Hamming distance versus the amount of power generated that allows CPA to build its correlations and extract the secret information.

B. Correlated Noise Generator

Since the key idea of statistical power analysis attacks is for the attacker to build a correlation between Hamming distance and power consumed, we designed the correlated noise generator to obfuscate that relationship by creating additional bit transitions simultaneously with the encryption/decryption. This obfuscation is demonstrated in Figure 2(b). While similar

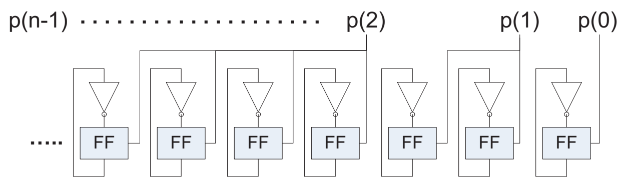


Figure 3: Power injector architecture that can toggle up to 2^n-1 nets each cycle by varying the power input p.

to Figure 2(a) in that it relates peak power during round one of DES to the Hamming distance produced by the model, we simulated correlated noise and added it to the power traces.

Our approach does not use random noise, which is ineffective because the attacker could simply filter the noise. Instead, the correlated noise generator uses the plain- or cipher-text as a seed to correlate the bit-switching to the encryption/decryption of the same data. This strategy makes the noise repeatable and harder to filter out.

During the first clock cycle, the seed is reduced to the same bit width as the power-injector input. A transformation is then applied to the seed, consisting of bit rotations and permutations, to generate an altered value. This altered value is then exclusive ORed with a random number generated at synthesis, referred to as the salt. The salted output is used as the input to the power injector. At the same time, the salted output is fed back into the transform to be used in place of the reduced seed for the remaining clock cycles. This system may be reset at any time to return to its initial state.

Although we evaluate correlated noise in defending the encryption key used for DES, the technique potentially applies to other statistical power analysis attacks because our method targets the linear relationship the attacks depend on. Additionally, the variable seed input means this technique can obscure the data leakage of other applications.

C. *Power Injector*

Whereas the correlated noise generator determines how power should be varied to obscure leakage, the power injector is responsible for generating that power. To accomplish this goal, we designed a register-transfer-level power-injector core that can be integrated into any overlay to toggle any number of nets every cycle. In addition to increasing power by specified amounts, the injector can also decrease power by using a baseline that includes some amount of toggling every cycle that can be increased or decreased each cycle.

Figure 3 demonstrates the architecture of the power injector. The injector takes as input an n-bit power setting p that specifies how many nets should be toggled. The value of n is specified pre-synthesis based on the application-specific required range of power, but p can change every cycle using power values from the correlated noise generator. To implement the toggling, the architecture uses 2^n-1 flip-flops with invertors. The enable for each flip flop is controlled by p, where bit i of p enables 2^i different flip flops. With this architecture, an overlay can change the number of toggled nets from 0 to 2^n-1 every cycle.

III. Experiments

In this section, we present a DES case study on the effectiveness of correlated noise against statistical power analysis as an example of a countermeasure that could be integrated into potentially any overlay. Although this case study demonstrates CPA against the first round of DES, our approach could be applied to both other forms of statistical analysis and to other applications requiring protection against such an attack.

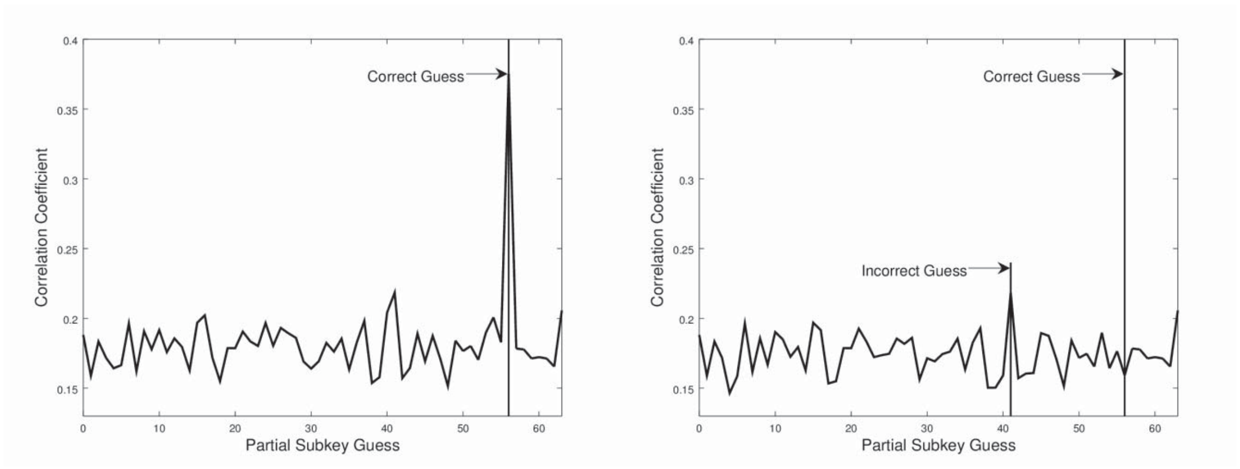
For this paper, we applied CPA to publicly available DES traces supplied by [6]. Each trace consists of 20,000 data points that have been pre-processed such that all rounds occur at the same time across traces. The experiments were performed using CPA with 500 traces. The operation we attacked was the Feistel function in round one, specifically the individual substitution boxes (S-boxes). Using the known plain text, we used the right-side values that are fed into the Feistel function as the known state and the right-side values after the round as the output of our model. The correlation factor is produced using a Pearson Correlation factor as described in [1].

The experiment was done in two stages. The first stage involved performing a CPA attack on the provided traces to act as a baseline, showing the effectiveness of the attack in general at breaking a single S-box. The second stage performed the same attack on power traces that we altered to simulate the effects of correlated noise as described in Section II.B. The goal was to demonstrate how correlated noise can obfuscate the correlation between Hamming distance and measured voltage. Only one S-box was broken, as the concept is equivalent across all S-boxes.

To perform the attack, we developed a Matlab function that modeled the first round of DES and returned the Hamming distance for the 4-bits affected by the specified S-box. We then used a wrapper function that tested all 64 possible partial subkey combinations and collected the respective Hamming distances. After we generated the Pearson coefficients correlating the Hamming distance and power traces, we observed the peak values that occurred during round one for each key guess. The highest coefficient was returned as the best guess. The peak coefficients for each guess can be seen in Figure 4(a). Notice that the correct guess, 56, has a distinctively high correlation compared to the other points.

To evaluate our countermeasure, we generated altered traces to mimic correlated noise generation by first determining the average effect of a single bit changing. We then averaged 3000 traces together to generate a general, representative trace. The representative trace was segmented into individual rounds and the segments normalized to the voltage level of a single bit change. For each trace used in the attack, we passed the plain-text into a simulator which determined the appropriate CNG output. In place of the power injector, we calculated the correct number of bit transitions the power injector would create and multiplied the normalized round segments by that number. We then added the segments to the power trace being altered.

Figure 4(b) demonstrates that when performing CPA on the altered traces, our countermeasure obfuscates the correlation between Hamming distance and power consumption when



(a) **(b)**

Figure 4: A successful CPA attack on a DES S-box (a) will produce a distinctively high correlation with the correct guess. However, when the correlation between the model used in the attack and the power traces is obfuscated (b), there is no longer a distinctive correlation with any guess, let alone the correct one.

using 500 traces. The figure shows the peak correlation value for each of the 64 guesses using the altered traces. As before, the highest correlation should occur at a guess of 56. However, we see no significant correlations formed with the highest peak occurring at guess 41. This peak renders the attack inconclusive, protecting the secret key and power consumption. This protection only held for up to 5000 power traces, which is typical of noise-based countermeasures, but this strategy can be applied to more-effective countermeasures.

Although space constraints prohibit a detailed analysis of the power-injector core, we provide a summary of area requirements and power-injection granularity. Because most FPGAs provide lookup tables (LUTs) with two flip-flops, and because the input to the flip-flop is just a one-input invertor, each LUT can implement two toggling flip-flops. For an n-bit power input, most FPGAs will require 2^{n-1} LUTs. The value of n will vary for different applications. Generally, the larger the value of n, the lower the SNR and in turn the more protection is granted. Therefore, a designer should pick the largest n that meets their constraints.

IV. Conclusions

Statistical power analysis is a significant hardware security concern for FPGA applications. Whereas previous work has focused on encryption cores that protect against such analysis, our approach complements those studies by providing an overlay-based countermeasure strategy that can protect widely used vulnerable cores and devices. We evaluate our approach by integrating correlated noise generation and power injection into FPGA overlays to obscure leakage of secret information by obfuscating the relationship between Hamming distance and voltage. We demonstrated a proof of concept by performing a correlation power analysis attack on publicly available DES power traces, and then demonstrated that our countermeasure mitigates the attack by yielding incorrect encryption keys. Future work includes integrating this, and more-effective, countermeasures into existing FPGA overlays and extending the approach to protect FPGA bitfiles by implementing the countermeasures in a partially reconfigurable region.

Acknowledgments

We gratefully acknowledge help from Draper, Swarup Bhunia, Domenic Forte, Jungmin Park, and Kai Yang.

References

[1] E. Brier, C. Clavier, and F. Olivier. *Correlation Power Analysis with a Leakage Model*, pages 16–29. Springer Berlin Heidelberg, 2004.

[2] D. Capalija and T. S. Abdelrahman. A high-performance overlay architecture for pipelined execution of data flow graphs. In *2013 23rd International Conference on Field programmable Logic and Applications*, pages 1–8. IEEE, 2013.

[3] P. Cooke, J. Fowers, G. Brown, and G. Stitt. A tradeoff analysis of fpgas, gpus, and multicores for sliding-window applications. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 8(1):2:1–2:24, Mar 2015.

[4] J. Coole and G. Stitt. Fast, flexible high-level synthesis from opencl using reconfiguration contexts. *IEEE Micro*, 34(1):42–53, Jan 2014.

[5] J. Coole and G. Stitt. Adjustable-cost overlays for runtime compilation. In *Field-Programmable Custom Computing Machines (FCCM)*, pages 21–24, May 2015.

[6] DPA Contest. http://www.dpacontest.org/home/.

[7] T.Güneysu and A. Moradi. Generic side-channel countermeasures for reconfigurable devices. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* , pages 33–48, 2011.

[8] W. Hnath. Differential power analysis side-channel attacks in cryptography. Worcester Polytechnic Institute, 2010.

[9] N. Kamoun, L. Bossuet, and A. Ghazel. Correlated power noise generator as a low cost dpa countermeasures to secure hardware aes cipher. In *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*, pages 1–6, Nov 2009.

[10] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proceedings of the International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, 1999.

[11] W. Luis, G. R. Newell, and K. Alexander. Differential power analysis countermeasures for the configuration of sram fpgas. In *IEEE Military Communications Conference (MILCOM)*, pages 1276–1283, Oct 2015.

[12] S. Skorobogatov and C. Woods. In the blink of an eye: There goes your aes key. *IACR Cryptology ePrint Archive*, 2012:296, 2012.

[13] G. Stitt, R. Karam, K. Yang, and S. Bhunia. A uniquified virtualization approach to hardware security. *IEEE Embedded Systems Letters*, PP(99):1–1, 2017.