Black-Box Side-Channel Attacks Highlight the Importance of Countermeasures
– An Analysis of the Xilinx Virtex-4 and Virtex-5 Bitstream Encryption Mechanism –

Amir Moradi, Markus Kasper, and Christof Paar

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
`{moradi, mkasper, cpaar}@crypto.rub.de`

Abstract. This paper presents a side-channel analysis of the bitstream encryption mechanism provided by Xilinx Virtex FPGAs. This work covers our results analyzing the Virtex-4 and Virtex-5 family showing that the encryption mechanism can be completely broken with moderate effort. The presented results provide an overview of a practical real-world analysis and should help practitioners to judge the necessity to implement side-channel countermeasures. We demonstrate sophisticated attacks on off-the-shelf FPGAs that go far beyond schoolbook attacks on 8-bit AES S-boxes. We were able to perform the key extraction by using only the measurements of a single power-up. Access to the key enables cloning and manipulating a design, which has been encrypted to protect the intellectual property and to prevent fraud. As a consequence, the target product faces serious threats like IP theft and more advanced attacks such as reverse engineering or the introduction of hardware Trojans. To the best of our knowledge, this is the first successful attack against the bitstream encryption of Xilinx Virtex-4 and Virtex-5 reported in open literature.

1 Introduction

The market for digital electronics is highly competitive. Thus, a new product has to provide a unique selling point to be successful. It can be technologically superior to other products, or convince with a better design, better quality, or appealing price. But having invested a lot of efforts and money into the development of a new product, which options does a company have to prevent competitors from stealing or even cloning the product? On the legal side, patents offer a handle to stem against many kinds of product piracy threats. However, patent registration and monitoring is an expensive task that comes with many pitfalls. Furthermore, patents expose know-how to competitors, and patent disputes often imply a high financial risk. Besides that, not all technology qualifies for patent registration. In this case a manufacturer only receives legal protection within the scope of copyrights. At the end of the day the throughout protection of intellectual property (IP) remains a task that needs to be considered at a technological level.

In our work we analyze the provided IP protection mechanism of Xilinx's Virtex-4 and Virtex-5 Field Programmable Gate Array (FPGA) families called bitstream encryption. We provide a detailed description of this real-world side-channel analysis, illustrating the steps required to perform a black-box analysis of a mostly undocumented target, i.e., the embedded decryption module. Our results provide many practical insights that are of avail for practitioners in industry and academia. They will learn to judge the feasibility of possible side-channel analyses and to evaluate the black-box side-channel security of electronic devices in a realistic attack scenario.

1.1 Content of this Paper

The paper is organized as follows. Next, we give a short overview of FPGAs and their security features. Here we mainly focus on Xilinx's bitstream encryption solution employed in the analyzed Virtex FPGAs and the consequences of a successful bitstream extraction. In Section 4 we provide an introduction to the employed attack method and detail the practical issues we encountered to mount our attack. Then, in Section 5 we have a short glance on the computing architecture followed by our experimental results and the final conclusion.

2 Introduction to FPGAs

Designers of digital embedded systems have three different technology options to choose from. These are application specific integrated circuits (ASICs), microcontrollers and FPGAs. ASICs are specially-tailored pieces of silicon that realize exactly the desired functionality. They provide the highest performance and are very cost efficient when produced in large quantities. On the other hand, ASICs implement a static functionality that can not be modified or updated once produced. Microcontrollers are a class of silicon devices that implement a fixed instruction set allowing the designer to write pure software programs. This offers a good flexibility and allows for updates of devices in the field. On the other hand, a microcontroller's performance is limited by its nature of sequential instruction processing. The third option available to designers, the FPGAs, close the gap between powerful but inflexible ASICs and highly flexible but performance-limited microcontroller-based solutions. An FPGA is an integrated circuit that consists of many configurable logic blocks (CLBs), which can be configured to represent basic digital design elements as, e.g., logic gates and registers. In addition, the connections inside an FPGA are configurable such that inputs and outputs of several building blocks can be connected to each other.

Similar to an ASIC, a designer of a digital system provides a hardware description language (HDL) representation of the digital design that the FPGA should implement. Modern FPGAs come with additional built-in functionality, such as RAMs, adders and multipliers or even complete microcontroller cores that can be included into the design. Instead of mapping the design to transistors in silicon to manufacture ASICs, the development tools for FPGAs will map the HDL representation to CLBs and routed connections. Developing for an FPGA is in fact very similar to the development of a microcontroller software. The difference is that the software will be executed by the microcontroller in a sequential fashion while the configuration of an FPGA allows for highly parallel designs. The software equivalent for FPGAs is called *bitstream* or *configuration*. There are multiple ways to store the configuration for an FPGA. Since their first generation, FPGAs use SRAM (Static Random Access Memory) to store the configuration. This requires that the FPGA has to be reconfigured after each power loss. In this setup, the configuration data is stored in an external non-volatile ROM (Read Only Memory) and is loaded on each power-up. Although today there are also devices that provide an internal Flash, EPROM, EEPROM or even one-time programmable memory (fuses), the market is still dominated by FPGAs using an external configuration ROM.

Being in general re-programmable, FPGAs offer the same flexibility as microcontrollers. On the other hand, they can achieve a much higher performance enabling many new applications that otherwise would require the power of an ASIC. FPGAs allow a fast time to market and many design iterations within a few hours. Therefore, FPGAs are often also used to test the digital functionality of ASIC designs in form of FPGA prototypes. Overall, FPGAs are more expensive than ASICs in medium to high volumes.

3 FPGA Security

Coming back to the initial question of IP protection we now introduce the general vulnerabilities arising from the existence of machine readable configuration files of FPGA designs.

3.1 Bitstream Vulnerabilities

During power-up, an SRAM-based FPGA reads its configuration from an external non-volatile memory. The configuration includes the functional design as well as the I/O configuration for the pins and the exact placement and routing of all used components. The whole design of an FPGA application is encoded within the configuration file, the role of which can be considered similar to the role of software for microcontrollers. As introduced before, this nature provides a means to update the configuration file of an FPGA to adapt its behavior to new system requirements or to fix early design flaws.

On the other hand, it also gives rise to the fact that the bitstream needs to be considered as the key element of an FPGA design and thus requires protection. In this section we discuss the impact of attacks on an unprotected bitstream and provide a glance at the broad scope of possible consequences.

Consider a company that just released a new product. An adversary will have easy and anonymous access to hardware, once the product is released. Thus, it seems reasonable to assume that a determined adversary will be able to find a way to access the plain configuration file, unless it has been protected by means of IP protection mechanisms. For now we ignore any possible protection and restrict our discussion to unprotected bitstreams. The bitstream extraction could be performed by eavesdropping the configuration process, unsoldering the configuration ROM or downloading a firmware update that includes a new FPGA bitstream. An adversary who stole a bitstream file can copy it and thus steal IP (Intellectual Property). This opens doors for product cloning and product piracy. The pirated products could be brought to the black-markets to earn money, without the need for the adversary to invest in product development. This causes damage in several ways. The IP-owning company would suffer from the lost sales directly due to the loss of income. But furthermore, the pirated products may be of weak quality and thus pose the additional threat to cause image loss. Even more, the cloned products might also come with additional functionality (e.g., offering a remote control) or a fancy optical design, such that there could even be seriously competing products originating from pirated hardware. In this case it might be hard to prove or even find that IP has been stolen and cloned.

Cloning FPGAs has even more serious implications in security sensitive scenarios, e.g., military technology like nuclear warheads or surveillance satellites. Adversaries that are able to extract the FPGA configuration would get access to highly sensitive technology. The possibility of bitstream reversal and manipulation that are discussed next, makes these scenarios even worse.

For many years people challenged the bitstream's security with respect to bitstream reversal. The motivations to do this are manifold. Some want to develop their customized toolchain for FPGAs and thus need to be able to compile an HDL design to a valid configuration file. Others are security researchers that search to extract secret algorithms or keys from the bitstream, and other parties might be interested in stealing a competitors technology. A good judgment of the difficulty to reverse engineer a bitstream is also essential for security evaluators and system designers that aim at increasing product security by selecting a suitable combination of secure devices and anti-tamper technologies to minimize the risk of vulnerabilities. Bitstream reversal can be used for proving infringement as well. These examples illustrate that both criminals and designers have a decent motivation to reverse engineer bitstreams.

The most studied bitstream format is that of the FPGA world market leader Xilinx, Inc. The Ph.D. thesis of Saar Drimer provides a good overview on FPGA security [4]. For lack of space, we only sum up some small parts of his discussion on bitstream reversal, and refer the interested reader to his work for all details. The general description of the structure of the bitstream can be found in Xilinx publicly available documents [18, 20–22]. Ziener *et al.* have shown [23] that the configuration of look-up tables (LUT) and RAM contents can be extracted from bitstreams with moderate efforts. According to Drimer, methods translating a bitstream to a netlist are not technically mature, yet. In open literature there are two works documenting notably successful reversals of bitstreams. The first is the free software project "Ulogic" by Note and Rannaud described in a report by the developers [12]. This work relates Xilinx Design Language (XDL) plaintext representations of placelists to bitstream bits, with a result that, as stated by the authors, is still a step away from a true netlist. The related "FPGA analysis tool" by Kepa *et al.* [7] adds a graphical representation to the decoded bitstreams and provides another step towards true reverse engineering. As the encoding of bitstreams is undocumented but not confidential by means of cryptography, it is a strong belief in industry and academia that bitstream reversal of FPGAs may be a difficult and time consuming, but nevertheless a technically-feasible task.

The possibility to reverse engineer a bitstream lets arise even further threats. Besides cloning and stealing IP, the reverse engineering of a bitstream also allows modifying the designs. This way Trojan Hardware could be added to a security-sensitive system. This malicious circuitry may, e.g., implement a hidden backdoor or some kind of kill-switch functionality to an FPGA that implements a sensitive application (e.g., nuclear power plant, military- or satellite technology or a banking application). Besides this, the option to modify a design also allows hobbyists to customize commercially available hardware to add functionality or improve performance. Beyond the discussion of the bitstream security, the interested reader is referred to [4] for a throughout evaluation of many aspects of FPGA security and to [2] where invasive attacks on FPGAs are discussed.

3.2 IP Protection for FPGAs[1]

As of 2001 [16], Xilinx implemented an encryption mechanism in many of its recent FPGA series released within the last decade to counter these threats. This mechanism is called *bitstream encryption* and works in the following way: instead of storing a plain bitstream file within the configuration ROM, the designer encrypts the bitstream configuration beforehand. The encryption – using AES-256 in CBC (Cipher Block Chaining) mode for the Virtex-4 and Virtex-5 FPGAs – is performed in software by the Xilinx ISE development tools. The used key is chosen by the designing engineer and is programmed into the Virtex FPGA. The part of the FGPA memory storing this secret key is battery-powered so that the key will immediately be erased on power loss of the battery support. This feature is designed to hinder invasive attacks to recover or reverse engineer a device configuration.

With the known encryption key inside the FPGA and the encrypted bitstream stored within a ROM, products can securely configure the FPGAs as only AES-256 encrypted data passes the channel between ROM and FPGA. The FPGA has a dedicated AES hardware to decrypt the bitstream. This hardware is not accessible for other purposes within the FPGA due to export regulations of cryptography.

3.3 Real-World Attacks

With this mostly theoretical discussion on several threats and countermeasures for FPGA applications, the remaining question is whether the manufacturer's countermeasures are able to provide the advertised protection in the real world, i.e., successfully prevent attacks. Unfortunately decisions on implemented countermeasures are in most cases driven by economical reasons. Thus, products in industry will only be guarded against risks and threats when customers are expected to be willing to pay for the additional security. Thus, often well-known security risks from academic literature are not considered when designing commercial products, as long as there is no evidence for real-world implications. One class of these attacks often underestimated in industry are the side-channel attacks introduced in the next section. In our contribution we analyze the IP protection mechanisms of recent FPGAs with respect to side-channel security. We show that the implemented features of the studied products can be broken with moderate efforts and thus fail to protect the implemented configuration file.

[1] Due to page restriction we limit this discussion to only the scheme provided by Xilinx FPGAs.

4 Side-Channel Analysis Attacks

4.1 Introduction to Side-Channel Analysis Attacks

Today Side-Channel Analysis (SCA) is a mature field in applied security research. Differential side-channel analysis methods have been introduced first by Kocher et al. around 10 years ago [8]. Since then the field has grown rapidly and many new tools and distinguishers for side-channel analysis have been evaluated. In reply to the new threat developed in the scientific literature many countermeasures have been proposed, implemented and broken. Also, experts from the field of theoretical cryptography recognized side-channel attacks as an important topic seeding a community of researchers working on general leakage resilience and provable security bounds for side-channel countermeasures. Beyond purely academic studies, side-channel attacks and reverse engineering have been shown to also have real-world impact. Examples are the attacks on NXP's Mifare Classic devices [11], a bouquet of attacks on Microchip's KeeLoq remote keyless entry systems (primary article [5]), and recently also SCA attacks on Mifare DESFire contactless smartcards [15]. Lately a successful side-channel key recovery attack on the bitstream encryption feature of the older Xilinx Virtex-II pro FPGAs, which employ 3DES as the decryption engine, has been reported in [10]. In this paper we describe a practical side-channel analysis attack on the bitstream decryption engines of the more recent Virtex-4 and Virtex-5 FPGAs. These attacks demonstrate that industrial products in fact require to implement side-channel countermeasures and that side-channel attacks are not a pure academic playground but have a real-world impact on the security of embedded systems.

The method employed in this work is a sophisticated type of Correlation Power Analysis (CPA) as first introduced in [3]. In this method the power consumption or electro-magnetic radiation (EM) of a device is measured while executing a cryptographic algorithm. In addition to the physical power consumption of the analyzed device, also the communication of the device is eavesdropped to get access to the ciphertexts (or plaintexts) that will be (or have been) processed. In our case the ciphertexts, i.e., blocks of the encrypted bitstream, are available by eavesdropping the configuration process and the analyzed cryptographic primitive is an AES-256 decryption module.

During the analysis itself the known ciphertexts are used to predict an intermediate value processed by the AES algorithm. A hypothetical intermediate value for each trace is calculated assuming a fixed subkey[2]. In the next step these hypothetical values are used in a hypothesis test, which allows distinguishing the key used by the device from wrong key hypotheses. In a CPA attack the used distinguisher is Pearson's correlation coefficient estimated by the sample correlation.

Side-channel analysis attacks follow a divide-and-conquer strategy. That is, the key is recovered in small pieces. Typical attacks use subkeys of 8 (AES) or 6 (DES) bits and target S-box outputs.

In our attack we can use a full bitstream as a set of multiple ciphertexts. In order to apply the correlation distinguisher, the predicted intermediate values have to be mapped to hypothetical power consumptions, which will then be compared with the measured

[2] By subkey we denote the part of a key that has an effect on the predicted intermediate value.

power consumption. For hardware designs a reasonable choice to do so is the Hamming distance (HD) model, which counts the number of bits of an intermediate value that are toggled within a clock cycle.

4.2 Measurement Setup

We have started our analysis by examining a Virtex-4 FPGA. We have used a “Virtex-4 FF668 Evaluation Board” [19], which provides a ZIF socket to host Virtex-4 devices with FF668 packaging. Since the board has not been designed for side-channel analysis, we have placed a resistor in the V_{CCINT} path and removed the blocking capacitors[3]. There are three different V_{CC} paths in Virtex-4 FPGAs: V_{CCINT} (1.2V) as the power pin for internal core circuits, V_{CCAUX} (2.5V) as the power pin for auxiliary modules, and V_{CCO} (1.2$\sim$3.3V) as the power pins for the output pin drivers. We analyzed all power pins, but similar to the results reported in [10], the successful results were obtained when considering the power traces measured in the V_{CCINT} path.

Our target Virtex-4 FPGA model was an XC4VLX25, and the power traces were captured using a LeCroy WP715Zi digital oscilloscope at a sampling rate of 2.5 GS/s and a LeCroy AP033 active differential probe. We have also designed a microcontroller based module which configures the FPGA in slave serial mode (see [20] for more details on Virtex-4 configuration modes). It communicates with a PC and passes the bitstream chunks[4] to the FPGA. The same board also provides a trigger pin to start the oscilloscope each time right before sending a bitstream chunk to the FPGA. The acquisition of power traces started after sending the header part of the bitstream. Each measured trace belongs to the previously sent bitstream chunk.

4.3 Introductory Experiments

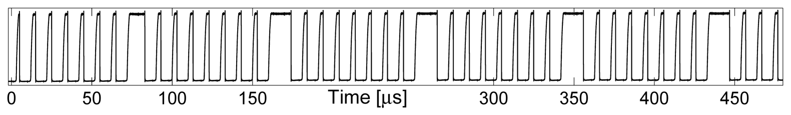
In a real world attack, it is of major importance to work very accurately and carefully to make sure the chosen method and all employed models are suited for the analysis. Thus, some preliminary work is required to eliminate uncertainties wherever possible. The first step in side-channel analysis is to find the correct instance in time when the targeted security primitive (here the AES-256 decryption) is processed. We created a very simple design using the Xilinx ISE development tools and generated both, the corresponding bitstream and its encrypted counterpart. The 128-bit IV (Initialization Vector)[5] and the 256-bit key used to generate the later one were loaded into the FPGA[6], whose V_{CCBAT} pin was continuously battery powered at 3.0V. Using the public documentation by Xilinx ([18, 21]) we verified the order of the bits and bytes of the ciphertexts within the encrypted bitstream.

[3] This task is essential and common when performing power analysis attacks on real-world devices, as the capacitors would filter the analyzed signal.

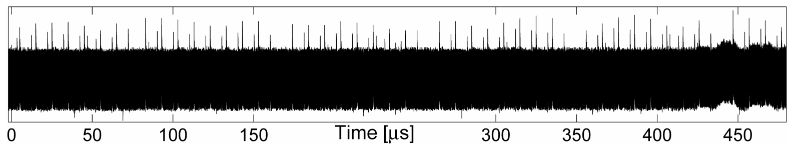
[4] Since the Virtex-4 bitstream encryption uses AES-256, we define each block of 128 bits as a chunk.

[5] The IV is used as initial value in CBC mode.

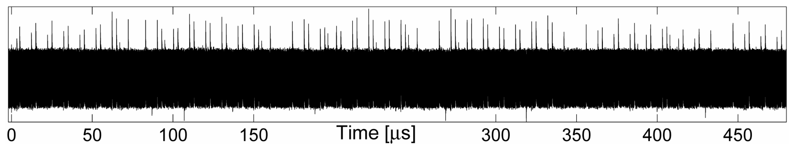
[6] Note that the only way to load the encryption key is through the JTAG interface [20] and by means of a standard configuration device.



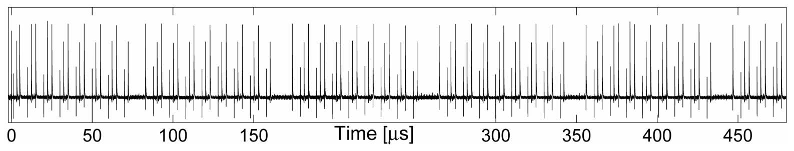
(a) Configuration clock signal



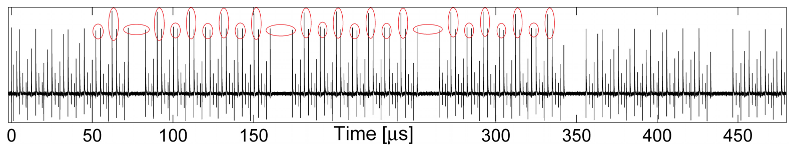
(b) A sample power trace, plain bitstream



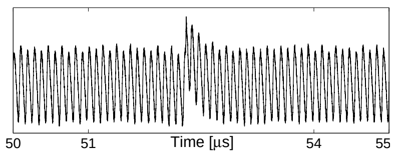
(c) A sample power trace, encrypted bitstream



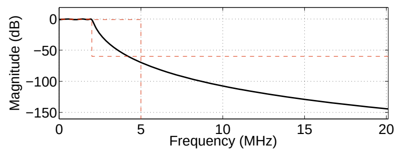
(d) A mean trace over 10 000 power traces, plain bitstream



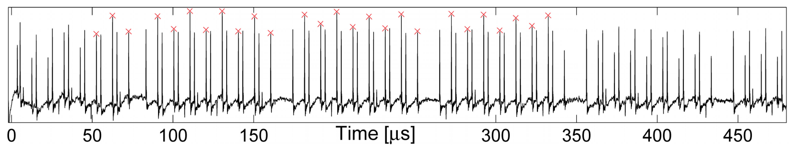
(e) A mean trace over 10 000 power traces, encrypted bitstream



(f) A part of a power trace, encrypted



(g) Magnitude Response of the used filter



(h) A sample filtered power trace, encrypted bitstream

Fig. 1. Virtex-4: Sample power traces, mean traces, and result of filtering

Comparing the power traces corresponding to the plain and the encrypted bitstreams let us to identify the interesting time instances. Two exemplary power traces are shown in Fig. 1(b) and Fig. 1(c). Due to the high level of noise present in the measurements we employed mean traces instead of raw measurement data in this step, i.e., each mean trace has been obtained by calculating the average of 10 000 traces. The mean traces, plotted in Fig. 1(d) and Fig. 1(e), show clear differences between the configuration with an active decryption module and the unencrypted configuration. We identified 26 clock cycles that show significant differences.

We should emphasize that in contrary to the Virtex-II case [10] (in which the full 3DES decryption is executed after a certain positive edge of the configuration clock signal[7]), the computations of the AES decryption rounds are spread and activated by consecutive positive edges of the corresponding configuration clock signal. Thus, in the case of the Virtex-4 the performance of the decryption module has much less impact on the maximum frequency of the configuration clock signal, as just single rounds need to be processed within a configuration clock cycle. According to the public documentation [17], when a Virtex-II is configured in SelectMAP mode using an encrypted bitstream, the BUSY signal has to be monitored[8] to ensure that the decryption module is ready for next data. The Virtex-4 FPGAs do not need to drive the BUSY signal during configuration, even when configuring using the maximum frequency and an encrypted bitstream [20]. In summary, our first experiment allowed us to find the most valuable instances in time for our SCA and gave us a hint towards a most likely round-based architecture.

A close look at a power trace (Fig. 1(f)) reveals that the measurements include HF-modulated waveforms. Therefore, we used a Chebyshev low-pass filter to reduce the effect of the high frequency components. The configuration of the used filter and the result of the filtering of a sample trace are shown in Fig. 1(g) and Fig. 1(h) respectively.

With this initial analysis and preprocessing of our measurements a remaining task was to find and verify a model for the internal architecture of the AES-256 decryption module, that allows us to relate the measured power consumption to the processing of the decryption primitive. The method that we used is to correlate the filtered power traces to predictions based on a hypothetical power model of the architecture. Thus, by trial and error, we guessed several possible architectures and modeled their power consumption. Using the known key we applied the models to the encrypted bitstream and correlated the resulting hypothetical power values to our measurements. In this experiment a significant correlation indicates a valid power model that might be a candidate to be used in the following cryptanalysis. Having examined several architectures and a couple of hypothetical power models, the only working model we found was the combination of the architecture shown by Fig. 2 and a HD model targeting the 128-bit register R.

Figure 3 shows the results of correlating the bit flips of the intermediate register between the first and second decryption rounds. More precisely, Fig. 3(a) has been obtained computing Pearson's correlation coefficient between each time instance of the

[7] TCK in the case of JTAG and CCLK in the other configuration modes

[8] The SelectMAP mode enables sending 8 bits of the bitstream at each clock cycle, and BUSY is one of relevant handshaking signals.

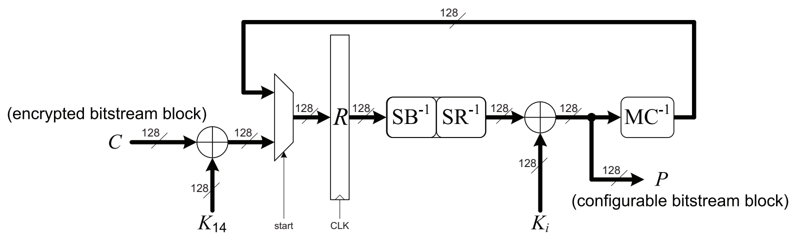


Fig. 2. The employed model of the internal architecture of the AES-256 decryption module

filtered power traces and HD of register R in the first and second decryption rounds, i.e., Hamming weight (HW) of

$$\Delta R_{1,2} = \Big[\underbrace{C \oplus K_{14}}_{R_1}\Big] \oplus \Big[\underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_1)\big) \oplus K_{13}\Big)}_{R_2}\Big],$$

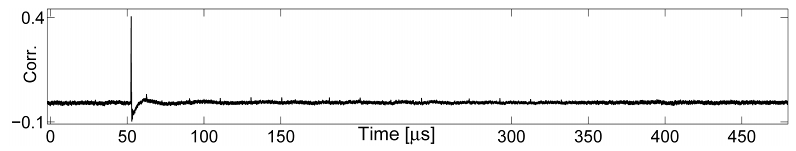
where C, K_{14} and K_{13} represent ciphertext, round key 14 and round key 13 respectively. Also, MC^{-1}, SB^{-1} and SR^{-1} are abbreviations for InvMixColumns, InvSubBytes and InvShiftRows transformations. The high peak at around 52μs indicates a very high dependency between the measured power traces and the intermediate values in our considered internal architecture. This time instance corresponds to the positive edge of the configuration clock signal at the sixth clock cycle (see Fig. 1(a)). In order to find which bit flips in register R causes the most significant correlation, we repeated the same computation considering each bit of $\Delta R_{1,2}$ independently. This led to the 128 curves shown in Fig. 3(b). For some yet unknown reasons, additional high peaks also appear in 13 other time instances. The results of applying a similar procedure in the next decryption round, i.e., using

$$\Delta R_{2,3} = R_2 \oplus \Big[\underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_2)\big) \oplus K_{12}\Big)}_{R_3}\Big],$$

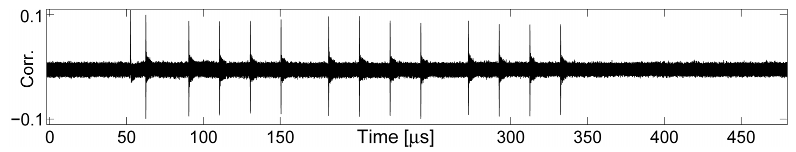
are depicted in Fig. 3(c). Notably here the high peaks only appear at one single time instance, i.e., 90μs corresponding to the start of the 9th configuration clock cycle. The curves in Fig. 3 have been derived using the power traces of the 60 000 decryptions performed during a single power-up of the FPGA. We should emphasize that no significant peak appeared when we continued this procedure for the next decryption rounds. In fact, it seems that our guess about the architecture does not completely match with the target internals. Nevertheless, according to the results these assumptions are adequate to successfully perform the attacks as shown later.

4.4 Implemented Attack

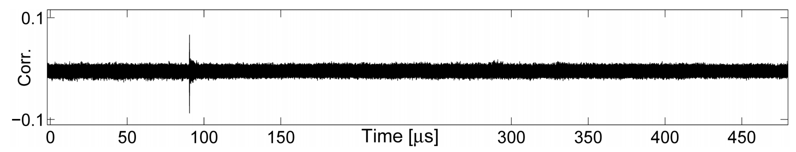
Using the extracted information about the leaking points and the architecture the straightforward way to perform an attack is to guess parts of two consecutive round keys K_{14}



(a) Using HD of full 128 bits of $\Delta R_{1,2}$



(b) Superposition of 128 curves considering each single-bit of $\Delta R_{1,2}$



(c) Superposition of 128 curves considering each single-bit of $\Delta R_{2,3}$

Fig. 3. Virtex-4: Results of correlating the filtered power traces to the bit flips in the intermediate register R

and K_{13}, and then use the single-bit power model to predict the key dependent leakage of the subsequent rounds. Due to the structure of InvMixColumns, at least one column (32 bits) of each round key has to be guessed at each step of the attack, which means searching the large key space of 2^{64}. However, because of the linear property of InvMixColumns one can write R_2 as

$$\underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_1)\big)\Big)}_{R'_2} \oplus \underbrace{\mathrm{MC}^{-1}\Big(K_{13}\Big)}_{K'_{13}}.$$

Moreover, since K_{13} and consequently K'_{13} are fixed and independent of the ciphertexts and decryption intermediate values, they can only change the polarity of our considered single-bit power model, i.e., bit flips of $\Delta R_{1,2}$. Therefore, guessing a column of K_{14}, i.e., searching a space of 2^{32}, is adequate when the single-bit power model is used. Note that in this case, one cannot take a power model using more bit flips, e.g., HD of whole 32 bits, in a CPA attack. However, a Mutual Information Analysis [6] or a multi-bit DPA [9] may be feasible.

We should highlight that one can decrease the search space of the attacks to the space of 2^8 by a chosen ciphertext scenario. For this, parts of R'_2 will be fixed as long as the corresponding ciphertext bytes are fixed. However, configuring the FPGA using a wrong encrypted bitstream (caused by the chosen ciphertexts) results in forbidden interconnections of internal wires, e.g., connecting two output pins to each other. Thus, each configured invalid bitstream block causes additional leakage currents, which lead to additional interferences with the measured side-channel signal. More importantly, these currents also heat up the FPGA and may even damage it. For this reason we did not pursue this approach and stuck to the approach using a valid encrypted bitstream. Nevertheless, when following the chosen ciphertext approach, the destructive effect of invalid bitstream chunks can be limited by resetting the configuration process after measuring a certain number of power traces, e.g., after each eighth chunk.

As a result, a full 128-bit K_{14} can be recovered by performing four attacks, each of which independently recovers a 32-bit part of the key. Note that in order to recover the full 256-bit key of AES-256, one needs to extract two consecutive 128-bit round keys, e.g., here K_{14} and K_{13}. We therefore need to extend the attack on the next decryption round. R'_2 can be computed for every ciphertext knowing K_{14}, and one can write

$$\Delta R_{2,3} = R'_2 \oplus K'_{13} \oplus \underbrace{\mathrm{MC}^{-1}\Big(\mathrm{SB}^{-1}\big(\mathrm{SR}^{-1}(R_2)\big)\Big)}_{R'_3} \oplus \underbrace{\mathrm{MC}^{-1}\Big(K_{12}\Big)}_{K'_{12}}.$$

As before, linear contributions of key bits, i.e., K'_{12} and K'_{13} can be omitted in our single-bit power model (here single bit flips of $\Delta R_{2,3}$). The attack described above can be run to recover the round key K_{13} which influences the hypotheses due to its contribution to R_2. Note that each part of this attack again recovers only a 32-bit column of K'_{13}. Knowing all bits of K'_{13} allows computing K_{13} by applying the MixColumns transformation. The result of the key extracting attacks and more details about their efficiency are given in Section 5.2.

4.5 Countermeasures

Today there exists a set of countermeasures that is believed to provide enough protection against side-channel attacks, that they can be considered secure for most practical purposes. More precisely, the reached level of security is boosted to a certain level which makes practical attacks not impossible, but infeasible in practice. Unfortunately most of these methods are patent-protected and thus often avoided in industry due to the involved royalties. Furthermore, many people in industry still recognize side-channel attacks as academic playground without any real-world impact and thus do not see the necessity of side-channel countermeasures for their products.

5 Implementing the Attack

5.1 Employing nVidia's CUDA

Our attack needs to perform an overall of eight analyses each statistically evaluating a set of 2^{32} key candidates. For each key candidate a hypothetical intermediate needs to be calculated for each used power trace. Fortunately, the locations of the occurring leakage we found earlier allowed us to limit the attack to a single time instance per decryption round. To cope with the large amounts of computations we employed NVidia's CUDA architecture[9] [14], to speed up our attack using the parallel computing capabilities of modern GPUs (Graphic Processing Unit). The used server was equipped with

[9] In the following we employ NVidias terminology of threads, blocks and grids as introduced in [14].

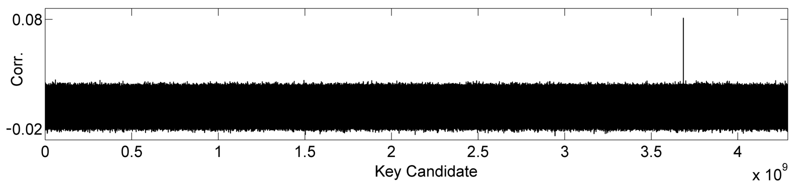


Fig. 4. Virtex-4: Result of an attack on the first column of the first decryption round using bit flips of the 7th LSB of the second byte

four NVidia Tesla C2070 cards [13], each having around 6 GB of memory and 448 thread processors arranged as 14 streaming multiprocessors. The implemented kernel processed one key per thread and was launched using a granularity of 256 threads per block and a $(64, 256, 256)$ grid. The resulting 2^{30} 32-bit floating point correlation coefficients per card were then stored to the machine's hard drive for visualisation in, e.g., MATLAB. Using CUDA allowed us to perform our analysis on a single point of 60 000 filtered power traces at a speed of one column each 33 minutes, i.e., an overall runtime of the computations of 264 minutes for attacking all 8 columns of the first two round keys K_{14} and K'_{13}. The corresponding correlation coefficients for the full attack require 128 GiB of hard disk space.

5.2 Attack Results

Amongst the available 32 bits of the register R that are suited to attack a column of the analyzed round key, we have selected the one which shows the highest absolute correlation at the datapoint at 52μs (see Fig. 3(b)). More precisely, the seventh LSB of the second byte of each column was selected in our attacks on the first decryption round. The result of the attack on the first column of the first round is shown in Fig. 4. Using 60 000 measurements, the highest correlation of 0.081 for the correct key candidate can already be clearly distinguished from the wrong key candidates. The next highest correlation value is already as low as 0.025. The attack on the second decryption round was performed in exactly the same way; even the same target bit was selected for the power model. As the results of the analysis of the second round closely reflect the results provided for the first decryption round, we refrain from providing extra figures at this point.

As mentioned before, we have used the measurements corresponding to only one power-up of the FPGA. The amount of possible measurements of each power-up depends on the size of the FPGA fabric (not to the used-defined design). The "Configuration Array Size" of the FPGA [20] defines how many 32-bit words have to be configured by a bitstream. Since 243 048 configurable words are available in our target, 60 762 traces can be measured using a single power-up. From the smallest Virtex-4 FPGA, XC4VLX15, 36 900 traces can be acquired during one power-up. Note that in the case of a high noise level the measurement process can be repeated with the same encrypted bitstream, i.e., with the same ciphertext values, and therefore provide more traces if required.

5.3 Differences to Virtex-5

We have examined the decryption module of a Virtex-5 FPGA re-employing the introduced analysis developed for the Virtex-4 device. The targeted FPGA model was an XC5VLX50 embedded on a SASEBO-GII [1]. We were able to reuse the measurement setup introduced before with minor modifications: Since the serial configuration pins used for the Virtex-4 were not present on the SASEBO-GII board, our microcontroller module was adapted to support configuration via JTAG interface, which is the only available configuration port on the used platform.

Compared to Virtex-4, the main difference was that the attack on the Virtex-5 FPGA required more power traces to be successful, which is mostly due to a worse signal-to-noise ratio due to a newer process technology (i.e., 65 nm instead of 90 nm). In our attacks we have used 90 000 traces[10] acquired during a single power-up of the FPGA, but using more traces of multiple power-ups can still improve discriminability of the correct key hypothesis. To deal with the worse measurement conditions, we have acquired power traces with a sampling rate of 20 GS/s and low-pass filtered the data as before. Approximately the same results as Fig. 1 were obtained, i.e., comparing the mean traces of the plain and encrypted bitstreams showed differences in the same positive edges of the configuration clock signal. Also, correlating the filtered power traces with the single-bit power model considering the same internal architecture led to the similar results shown in Fig. 3 but with lower correlation value, i.e., 0.05 and 0.03 for the first and second decryption rounds respectively. The analysis runtime increased due to the higher number of power traces to around 49 minutes per column or around 6.5 hours for the overall computing time. We should emphasize that analyzing the pure Virtex-5 without having the knowledge obtained during the analysis of Virtex-4, would have been a much more challenging task with more uncertainties.

6 Conclusion

Today, industrial spying and technology theft are a major threat for both companies and government-run facilities. Companies are mostly concerned about IP theft and product piracy and the inflicted losses. Government institutions, on the other hand, need to protect military secrets as well. Our attacks show that the IP protection mechanism of the FPGA world market Xilinx, Inc. can be circumvented using moderate efforts.

Our presented approach allows us to read out the configuration data of Virtex-4 and Virtex-5 devices in the field, leading to the consequences elaborately discussed in Section 3.1. Manufacturers of high-security products and security evaluation labs are well aware of the side-channel vulnerabilities. Therefore, they ensure that additional security countermeasures to protect devices are implemented where necessary. Techniques include for example shielding and molding the electronic circuit to provide additional tamper resistance and therefore deny power or EM measurements. Unfortunately this awareness does not cover all manufacturers of security sensitive devices yet.

We want to underline that this attack targets Xilinx's bitstream encryption engine, and not a third party crypto-implementation inside an FPGA. Although reading out and

[10] Our Virtex-5 target FPGA allows for measuring 98 031 traces during a single power-up.

interpreting the bitstream might also annihilate the security targets of an FPGA design, there is an important difference between a vulnerability in the bitstream encryption and a vulnerability in an implemented primitive. An engineer developing an FPGA design has no influence on the security of the bitstream encryption and thus also no option to improve it. In other words, up to now it is the FPGA manufacturer's responsibility to provide secure IP protection methods. This is slightly different to the microcontroller scenario. Designers using microcontrollers often have the freedom to implement customized bootloaders, that might, e.g., add encryption functionality to the programming. Nevertheless, the microcontroller's manufacturer also has to ensure that the memory including the bootloader and all its secrets cannot be read out.

There are many new insights from this attack. This is the first case to our knowledge, where it was possible to probe and compare the security of subsequent technology generations of an embedded system in a real-world environment. In this attack we were able to practically verify that an attack on more recent technology nodes still scales within feasible bounds. Furthermore, we were able to show that developing an attack tailored to one product can threaten the security of another product, when a security design is being reused. In our case the analysis of the Virtex-4 allowed us to study the architecture with much less efforts than having to perform the same analysis on the Virtex-5 device. In consequence, we suggest to limit the reuse of security designs, such that the security of a newer product is not lowered by an easier attack on an older product.

Another argument we practically disproved is that attacks on intermediate values that require large key hypotheses are infeasible in practice. We have shown that with todays available computing power an analysis on 60 000 power traces using 32-bit key hypotheses can be performed in less than 4.5 hours. We also explained the different steps that needed to be done to execute a black-box analysis. These differ from purely academic studies, as they include many additional steps as identifying and filtering unknown additional noise sources, the identification of the time instances that need to be considered in the attack and the deduction of a valid model of the implemented architecture. To our knowledge there is no published real-world side-channel attack with a similar attack complexity. Therefore, this work provides an update on the lower bound of attacks that should be considered a realistic threat to real-world systems.

The presented approach practically illustrates that side-channel attacks on real-world systems do not require any detailed knowledge of the implemented architecture. Thus, the extend to which confidential details on the implemented architecture can raise the difficulty of black-box side-channel analyses should not be overestimated. Nevertheless, it remains an open research problem to evaluate the security gain achieved by applying additional confidential obscurity measures as transformations of the plaintexts prior to encryption.

Finally, the most exciting question is to ask why this attack was possible at all. Side-channel analyses are known for more than ten years, and the same holds for bitstream encryption protection mechanisms. Why did Xilinx not implement the available countermeasures? As stated before, it is most likely due to an economic reason. In this case the FPGA configuration has been protected by means of an encryption mechanism. Customers accepted the solution without having the expertise to recognize the obvious possibility of side-channel attacks, and thus did not give rise to a market demand for a side-channel resistant configuration solution. The fact that Xilinx's bitstream encryption has not been broken in public literature for around a decade shows that side-channel attacks on real-world targets, i.e., black-box attacks, just became mature within the last years. From our point of view a prominent problem in security technology is that both, customers and manufacturers, are not aware of the security risks that come with unprotected implementations of cryptographic primitives in embedded systems. On the other hand, those that are aware of the existence of side-channel attacks, often consider them as a purely academic threat without any real-world counterpart. We see an urgent need to change this wrong perception and to recognize the rapid advances in the minatory field of black-box side-channel analysis. A general guideline should be that cryptographic routines in embedded systems without SCA countermeasures should be considered insecure for all applications where a successful attack can give rise to financial benefits.

To assist hardware manufacturers, scientific research should in the future aim at developing mechanisms beyond mere SCA resistance to face the increasing threat of physical attacks. This could be, e.g., protocols and measures that limit the effect of successful side-channel attacks. In the case of the bitstream encryption a solution avoiding repetitive use of the same key in CBC mode, e.g., by means of some obscure key transformation, would have significantly hardened the analysis. In this context researchers should also reconsider to add obscurity measures in combination with the well-proven crypto primitives to raise the SCA protection of their systems. This would require an attacker to first overcome an additional reverse-engineering step, before being able to analyze the system.

7 Acknowledgements

The authors want to thank Alessandro Barenghi from Politecnico di Milano for providing the programming tools needed to transfer our attacks to the Virtex 5 FPGA family. Furthermore, we want to thank David Oswald for many fruitful discussions and his advices on the Fourier preprocessing.

The work described in this paper has been supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II and by the German Federal Ministry of Education and Research BMBF (grant 01IS10026A, Project EXSET).

References

1. Side-channel Attack Standard Evaluation Board (SASEBO-GII). Further information is available via `http://staff.aist.go.jp/akashi.satoh/SASEBO/en/board/sasebo-g2.html`.
2. A. Braeken, S. Kubera, F. Trouillez, A. Touhafi, N. Mentens, and J. Vliegen. Secure FPGA Technologies and Techniques. In *FPL 2009*, pages 560–563. IEEE, 2009.
3. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.
4. S. Drimer. *Security for volatile FPGAs*. PhD thesis, Computer Laboratory, University of Cambridge, United Kingdom, 2009.

5. T. Eisenbarth, T. Kasper, A. Moradi, C. Paar, M. Salmasizadeh, and M. T. M. Shalmani. On the Power of Power Analysis in the Real World: A Complete Break of the KeeLoq Code Hopping Scheme. In *CRYPTO*, volume 5157 of *LNCS*, pages 203–220. Springer, 2008.
6. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In *CHES - 2008*, volume 5154 of *LNCS*, pages 426–442. Springer, 2008.
7. K. Kepa, F. Morgan, K. Kosciuszkiewicz, L. Braun, M. Hübner, and J. Becker. FPGA Analysis Tool: High-Level Flows for Low-Level Design Analysis in Reconfigurable Computing. In *ARC 2009*, volume 5453 of *LNCS*, pages 62–73. Springer, 2009.
8. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
9. T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard*, pages 151–161. USENIX Association, 1999.
10. A. Moradi, A. Barenghi, T. Kasper, and C. Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from xilinx Virtex-II FPGAs. In *CCS 2011*, pages 111–124. ACM, 2011.
11. K. Nohl, D. Evans, Starbug, and H. Plötz. Reverse-Engineering a Cryptographic RFID Tag. In *USENIX Security Symposium*, pages 185–194. USENIX Association, 2008.
12. J.-B. Note and É. Rannaud. From the Bitstream to the Netlist. In *FPGA 2008*, page 264. ACM, 2008.
13. NVidia. NVIDIA's Next Generation CUDA Compute Architecture: Fermi. `http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIAFermiComputeArchitectureWhitepaper.pdf`, 2009.
14. Nvidia. CUDA Developer Zone (Website), 2011. `http://developer.nvidia.com/category/zone/cuda-zone`.
15. D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *CHES 2011*, volume 6917 of *LNCS*, pages 207–222. Springer, 2011.
16. S. Trimberger. Trusted design in FPGAs. In *DAC 2007*, pages 5–8. ACM, 2007.
17. Xilinx, Inc. Virtex-II Pro and Virtex-II Pro X FPGA User Guide. `http://www.xilinx.com/support/documentation/user_guides/ug012.pdf`, 2002.
18. Xilinx, Inc. Application Note XAPP151 (v1.7), Virtex Series Configuration Architecture User Guide. `http://www.xilinx.com/support/documentation/application_notes/xapp151.pdf`, 2004.
19. Xilinx, Inc. Virtex-4 FF668 Evaluation Board. `http://www.xilinx.com/products/boards-and-kits/HW-AFX-FF668-400.htm`, 2004. User Guide: `http://www.xilinx.com/support/documentation/boards_and_kits/ug078.pdf`.
20. Xilinx, Inc. Virtex-4 FPGA Configuration User Guide. `http://www.xilinx.com/support/documentation/user_guides/ug071.pdf`, 2004.
21. Xilinx, Inc. Application Note XAPP138 (v2.8), Virtex FPGA Series Configuration and Readback. `http://www.xilinx.com/support/documentation/application_notes/xapp138.pdf`, 2005.
22. Xilinx, Inc. Virtex-5 FPGA Configuration User Guide. `http://www.xilinx.com/support/documentation/user_guides/ug191.pdf`, 2006.
23. D. Ziener, S. Assmus, and J. Teich. Identifying FPGA IP-Cores Based on Lookup Table Content Analysis. In *FPL 2006*, pages 1–6. IEEE, 2006.