Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series

Amir Moradi(✉) and Tobias Schneider

Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Bochum, Germany
{amir.moradi,tobias.schneider-a7a}@rub.de

Abstract. Since 2012, it is publicly known that the bitstream encryption feature of modern Xilinx FPGAs can be broken by side-channel analysis. Presented at CT-RSA 2012, using graphics processing units (GPUs) the authors demonstrated power analysis attacks mounted on side-channel evaluation boards optimized for power measurements. In this work, we extend such attacks by moving to the EM side channel to examine their practical relevance in real-world scenarios. Furthermore, by following a certain measurement procedure we reduce the search space of each part of the attack from 2^{32} to 2^8, which allows mounting the attacks on ordinary workstations. Several Xilinx FPGAs from different families – including the 7 series devices – are susceptible to the attacks presented here.

1 Introduction

Side-Channel Analysis (SCA) attacks have become a serious threat to cryptographic implementations. This indeed has been highlighted by publicly reporting several successful attacks on commercial devices, e.g., [1,5,9,14,15,20]. One of the well-known examples are the attacks on the bitstream encryption feature of FPGA devices which also garnered the attention of (industry and academic) FPGA communities.

The first SCA attack on the bitstream encryption of (out-dated and discontinued) Xilinx Virtex-II pro family has been presented in [11], where a full 168-bit key of the underlying triple-DES algorithm could be recovered by a single power-up of the FPGA (≈70,000 traces) by searching in a space of 2^6 for each 6-bit part of the key. The second work [12] showed that a similar attack on more recent Xilinx FPGA families (Virtex-4 and Virtex-5) is feasible. However, due to the underlying AES-256 algorithm and the implementation architecture, the presented attack could only recover the key by searching in a space of 2^{32} for each 32-bit part of the key. To deal with such a complexity, the authors made use of four graphics processing units (GPUs with a total of 4×448 thread processors) and mounted the attack on a single point of the 60,000 power traces collected from a single power-up of the FPGA. The full 256-bit key could be recovered in 4.5 h by such a setup while the attack on the second round (to recover the second 128-bit key) was not as efficient as that on the first round. In all the aforementioned attacks, power traces of various SASEBO or SAKURA boards have been collected. Since such boards are explicitly designed for power analysis evaluation purposes, remounting the same attacks on real-work applications might be challenging, where PCB should be slightly modified to provide a suitable measurement point.

As a side note, similar attacks on Altera FPGAs (Stratix-II and Stratix-II families) have been later reported in [13,17]. Compared to that on Xilinx FPGAs, the attacks required a reverse-engineering step (of the software development tools) and a sophisticated measurement procedure to deal with the underlying AES algorithm in counter mode.

Our Contribution. In this work we present an improved attack on bitstream encryption of modern Xilinx FPGAs. Our achievements can be summarized as follows:

- By further investigation of the design architecture of the AES decryption module, we present a more suitable power model for the attacks, particularly on the second cipher round.
- By means of a dedicated measurement setup, we reduce the search space from 2^{32} for each part of the attack to 2^8. Therefore, the attacks can be performed using ordinary desktop computers.
- We present the result of the attacks on Virtex-5, Spartan-6, Kintex-7, and Artix-7 FPGAs as the samples of 5, 6, and 7 series.
- In contrast to all reported attacks on Xilinx bitstream encryption, we present the results via electro magnetic (EM) side channel.

In short, we avoid the need of using GPUs, and demonstrate strong and efficient attacks on bitstream encryption of 7 series FPGAs of Xilinx which are currently in production.

2 Preliminaries

2.1 Xilinx Bitstream Encryption

Bitstream encryption, in general, has been introduced to prevent cloning and counterfeiting the user designs. In order to protect proprietary algorithms, secret materials, and obfuscated designs from reverse engineering, it is essential for the user to employ bitstream encryption. Xilinx products are mainly SRAM-based FPGAs, which implies reconfiguration (loading bitstream into the FPGA) every time the FPGA powers up. Since the bitstream has to be stored outside the FPGA (in a non-volatile memory), bitstream encryption is a must-to-have feature for the FPGA vendors, whose products are based on volatile memory (e.g., Xilinx).

The current available FPGA series of Xilinx make use of AES-256 in cipher block chaining (CBC) mode to encrypt the bitstream. Suppose that the bitstream is divided into n 128-bit blocks $p^{i\in\{1,\dots,n\}}$. The encrypted bitstream, which is formed by n 128-bit blocks c^i, is generated by

$$c^i = \text{AES}_k^{\text{ENC}}(p^i \oplus c^{i-1}),$$

assuming $c^0 = IV$. The secret key k and the initialization vector IV can be arbitrary selected by the user. The Xilinx development tools generate a human-readable ASCI file (with .nky extension) of the selected key and IV, which is given to the programming device to store the key inside the FPGA. As a side note, although IV is written into the .nky file, the programming device stores only the key into the FPGA via the JTAG port. Older versions of the Xilinx FPGAs make use of only volatile memory for the key storage which requires an external battery during power shortage. The newer families are, in addition, equipped with one-time programmable fuses.

Although there are not many public documents about the details of the structure of the bitstream file, with moderate efforts (similar to that of [11,12]) the essential information can be revealed (e.g., bit and byte endianness and the size of the header before the encrypted part starts). Such an investigation recovered that IV (in plain) is available in the bitstream before the encrypted part starts. Further, this IV must not be necessary the same as the one which has been formerly written into the .nky file.

2.2 Configuration and Measurement

The encrypted bitstream can be sent to the FPGA via several different protocols (serial, parallel, master, slave, and JTAG). Since the JTAG port is dedicated to configuration (and it has to be used for key programming), such a port is usually available in most of the real-world applications (e.g., set up boxes). In [12], a customized micro-controller (MCU) has been used to configure the FPGA (via JTAG) and provide a trigger signal for the oscilloscope. In [11,12], by monitoring the voltage drop of a resistor in VDD_{int} path, power consumption traces of the FPGA have been collected. The decryption module inside the FPGA receives 128-bit ciphertext blocks $c^{i\in\{1,\dots,n\}}$ in a consecutive fashion, and derives the plaintexts p^i as

$$p^i = \text{AES}_k^{\text{DEC}}(c^i) \oplus c^{i-1},$$

with $c^0 = IV$.

It has been reported in [12] that – in addition to the decryption engine – other modules of the FPGA are active whose energy consumption (as noise) are visible through the measured power traces. Hence, filtering the traces to reduce the noise was essential. As shown in Fig. 1, the decryption of c^i takes place when the next block c^{i+1} is fed into the FPGA. Further, the decryption clock is somehow synchronized with the JTAG clock.

2.3 Attack

Since AES-256 consists of 14 rounds, which fits to the 14 visible peaks in the power trace, it has been assumed that the decryption module in the FPGA realizes a round-based architecture of the AES-256, which performs one cipher round at each single clock cycle [12]. Figure 2 shows the hypothetical design architecture that has been considered in [12].

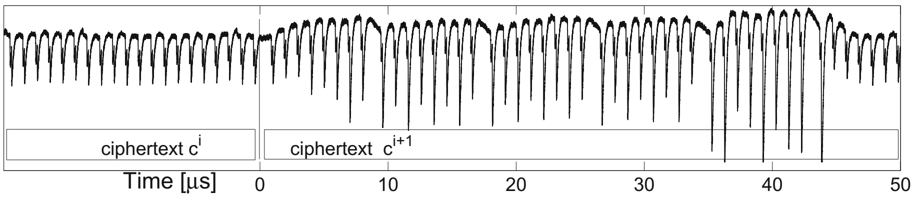


Fig. 1. A sample power trace of Spartan-6 (with 20 MHz low-pass filter) during loading an encrypted bitstream

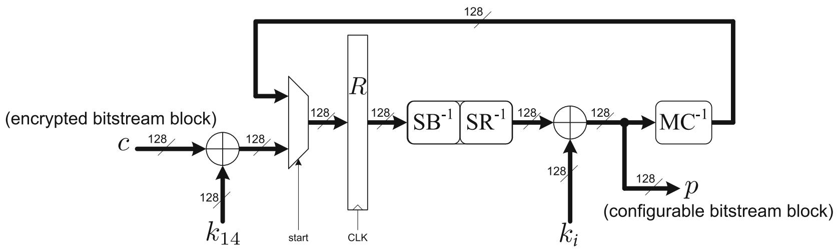


Fig. 2. Hypothetical design architecture of the AES-256 decryption module of modern Xilinx FPGAs (taken from [12])

Assuming such an architecture, the state register R stores $R_1 = c \oplus k_{14}$ and $R_2 = \mathrm{MC}^{-1}\Big(\mathrm{SR}^{-1}\big(\mathrm{SB}^{-1}\left(c \oplus k_{14}\right)\big) \oplus k_{13}\Big)$ at the first and second cipher rounds respectively[1]. In general, at round $1 < i < 15$ the content of the state register is $R_i = \mathrm{MC}^{-1}\Big(\mathrm{SR}^{-1}\big(\mathrm{SB}^{-1}\left(R_{i-1}\right)\big) \oplus k_{15-i}\Big)$. Indeed, the above shown hypothetical architecture has been verified by examining the correlation between the measured power traces and the Hamming distance (HD) of the state register in a known-key settings. As shown in Fig. 3, the power traces show a clear dependency on $\mathrm{HD}(R_1, R_2)$. However, such a dependency is strongly mitigated (but still available) in the next cipher round, i.e., on $\mathrm{HD}(R_2, R_3)$.

Let us denote $R_1 \oplus R_2$ by Δ_{R_1,R_2} and its byte at position $i \in \{0, \ldots, 15\}$ with $\Delta^{(i)}_{R_1,R_2}$. Following the AES notations, we represent the first column of the state R by $R^{(0,1,2,3)}$. Hence, due to the linear property of the MixColumns and

[1] MC: MixColumns, SR: ShiftRows, SB: SubBytes.

its inverse we can write

$$\begin{aligned}\Delta_{R_1,R_2}^{(0,1,2,3)} &= R_1^{(0,1,2,3)} \oplus R_2^{(0,1,2,3)} \\ &= c^{(0,1,2,3)} \oplus k_{14}^{(0,1,2,3)} \oplus \mathrm{M1C}^{-1}\Big(\Big\langle S^{-1}\left(c^{(0)} \oplus k_{14}^{(0)}\right), S^{-1}\left(c^{(13)} \oplus k_{14}^{(13)}\right), \\ &\quad S^{-1}\left(c^{(10)} \oplus k_{14}^{(10)}\right), S^{-1}\left(c^{(7)} \oplus k_{14}^{(7)}\right)\Big\rangle\Big) \oplus \mathrm{M1C}^{-1}\left(k_{13}^{(0,1,2,3)}\right), \qquad (1)\end{aligned}$$

where S^{-1} stands for the Sbox inverse, and $\mathrm{M1C}^{-1}$ for the inverse of the Mix-Columns operation on a single column.

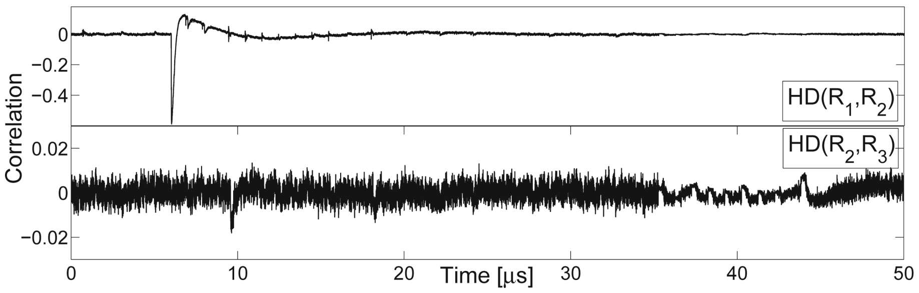


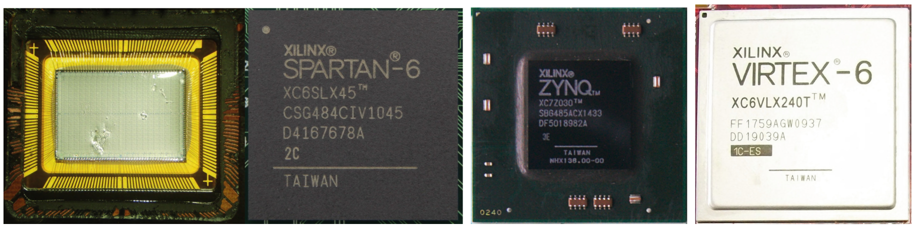
Fig. 3. Spartan-6, correlation between **power** traces and $\mathrm{HD}(R_1, R_2)$ and $\mathrm{HD}(R_2, R_3)$

Since both $k_{14}^{(0,1,2,3)}$ and $\mathrm{M1C}^{-1}\left(k_{13}^{(0,1,2,3)}\right)$ are fixed and independent of the ciphertext c, correlation power analysis (CPA) [2] (respectively classical DPA [7]) attacks, that target bits of $\Delta_{R_1,R_2}^{(0,1,2,3)}$, can be performed by guessing four key bytes $\left\langle k_{14}^{(0)}, k_{14}^{(13)}, k_{14}^{(10)}, k_{14}^{(7)}\right\rangle$. Such a 2^{32}-bit attack (on a single point of the power traces) has been performed in [12] using GPUs. The same attack with the same principle can be performed on the other columns of the Δ_{R_1,R_2} to recover full 128-bit round key k_{14}.

Having k_{14}, we can follow the same procedure for the second cipher round. Let us denote $\mathrm{MC}^{-1}\Big(\mathrm{SR}^{-1}\big(\mathrm{SB}^{-1}\left(c \oplus k_{14}\right)\big)\Big)$ by c' and $\mathrm{MC}^{-1}\left(k_{13}\right)$ by k'_{13}. As an example, for the first column of Δ_{R_2,R_3} we can write

$$\begin{aligned}\Delta_{R_2,R_3}^{(0,1,2,3)} &= R_2^{(0,1,2,3)} \oplus R_3^{(0,1,2,3)} \\ &= c'^{(0,1,2,3)} \oplus k_{13}'^{(0,1,2,3)} \oplus \mathrm{M1C}^{-1}\Big(\Big\langle S^{-1}\left(c'^{(0)} \oplus k_{13}'^{(0)}\right), S^{-1}\left(c'^{(13)} \oplus k_{13}'^{(13)}\right), \\ &\quad S^{-1}\left(c'^{(10)} \oplus k_{13}'^{(10)}\right), S^{-1}\left(c'^{(7)} \oplus k_{13}'^{(7)}\right)\Big\rangle\Big) \oplus \mathrm{M1C}^{-1}\left(k_{12}^{(0,1,2,3)}\right). \qquad (2)\end{aligned}$$

The same attacks (as shown in [12]) can target the bits of $\Delta_{R_2,R_3}^{(0,1,2,3)}$ and search in a space of 2^{32} to recover $\left\langle k_{13}'^{(0)}, k_{13}'^{(13)}, k_{13}'^{(10)}, k_{13}'^{(7)}\right\rangle$. The same procedure is repeated for other columns of Δ_{R_2,R_3}, and after revealing k_{14} and k'_{13} the 256-bit main key can be derived.



(a) (b) (c)

Fig. 4. Different packaging technologies: (a) wire-bond, (b) flip-chip, (c) flip-chip with lid-heat spreader

3 Our Analysis

3.1 Packaging

In contrast to all the reported SCA attacks mounted on bitstream encryption of Xilinx devices, we concentrate on EM analysis. Figure 4 shows two different packaging technologies flip-chip and wire-bond.In case of wire-bond, the metal layers (of the FPGA chip) are at the top side, and the bonding wires are covered by molding components (usually plastic, see Fig. 4(a)). Since the main EM radiations are due to the current flowing through VDD path(s), the EM probes can be placed at the top of the chip, if the top metal layers include the VDD_{int} (see Fig. 5(b)). For the flip-chip technology, sometimes the top of the chip is covered by a lid-heat spreader (Fig. 4(c)), which must be removed for EM analysis. Compared to the wire-bond case, the silicon side of the chip (usually a thick layer) is accessible, which prevents reaching the layers carrying VDD. Hence, the EM signals are usually weak unless the thick silicon is thinned by means of sophisticated polishing devices, that also allows using localized EM microprobes [6].

3.2 Measurements

For the EM measurements we used a digital oscilloscope at a sampling rate of 5 GS/s and bandwidth of 1.5 GHz. We have employed only near-field probes of LANGER EMV-Technik. Further, depending on the amplitude of the signal, we made use of one or two high-bandwidth AC amplifiers ZFL-1000LN+ from Mini-Circuits.

Depending on the packaging, type of the FPGA, and the visibility of the signal, we used either RF-U5-2 or RF-R50-1 EM probes. In case of Virtex-5 and Kintex-7 (both with flip-chip) as well as Artix-7 (wire-bond) we achieved the best results with a RF-R50-1 probe, and for Spartan-6 (wire-bond) with a RF-U5-2 probe (see Fig. 5). Except removing the lid-heat spreader of the Virtex-5 FPGA, we did not modify the packaging of the FPGAs.

For the sake of simplicity, we concentrate on the Spartan-6 case, and discuss the other FPGAs at the end of this section. We also developed a MCU-based device to configure the FPGAs through the JTAG port. Figure 6 shows a single EM trace of the Spartan-6 FPGA (synchronized with that of Fig. 1). As a proof of concept, and to verify the hypothetical design architecture, in a known-key scenario we measured 100,000 traces and estimated the correlation considering $\mathrm{HD}(R_i, R_{i+1}),\ 0 < i < 14$. The results, which are shown in Figure 6, indicate that the high correlation only exist at the first cipher round, which makes the attacks challenging at the second round.

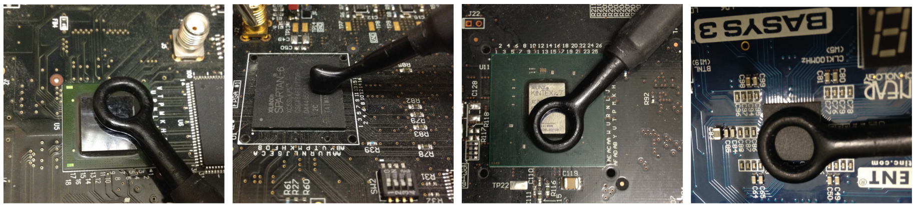

(a) Virtex-5 (b) Spartan-6 (c) Kintex-7 (d) Artix-7

Fig. 5. EM probes and different FPGAs, (a) XC5VLX50-1FFG324, (b) XC6SLX75-2CSG484C, (c) XC7K160T-1FBGC, (d) XC7A35T-1CPG236C

We have tried many different hypotheses for the design architecture, and finally the highest correlation has been observed considering the same architecture as shown in Fig. 2 but with $\mathrm{HD}(R_1, R_{i+1}),\ 0 < i < 14$ model. Although no design architecture can justify why the SCA leakage depends on the state register at round $i+1$ and that of the first round, such a model leads to considerably high correlations[2] as shown in Fig. 6.

The previous attacks have been based on measuring one or multiple power-ups of the FPGA [12]. This means that the ciphertexts have been previously defined (stored in a non-volatile external memory). Instead, we aim at selecting the ciphertexts by our choice. Sending chosen cipherexts to the FPGA, however, has a negative consequence on the interconnections of the FPGA. The switch boxes and look-up tables are wrongly configured which leads to short circuits (high power consumption and high temperature) and may destruct certain modules. Therefore, in order to avoid such consequences, after sending one (or a couple of) chosen ciphertext(s), the configuration process should be restarted. This can be easily done by sending certain commands through the JTAG port, which are available in Xilinx public documents, e.g., [18]. Following such instructions, we adjusted our MCU-based programmer to perform a configuration reset after each single measurement. In more details, after starting the configuration process the MCU device sends the header (the unencrypted part of the bitstream), the chosen ciphertext, and a dummy 128-bit ciphertext block. When the dummy ciphertext is sent, the corresponding EM/power trace is measured,

[2] As a side note, we found this leakage model by coincidence, and it is valid for all considered FPGAs and for both power and EM leakages.

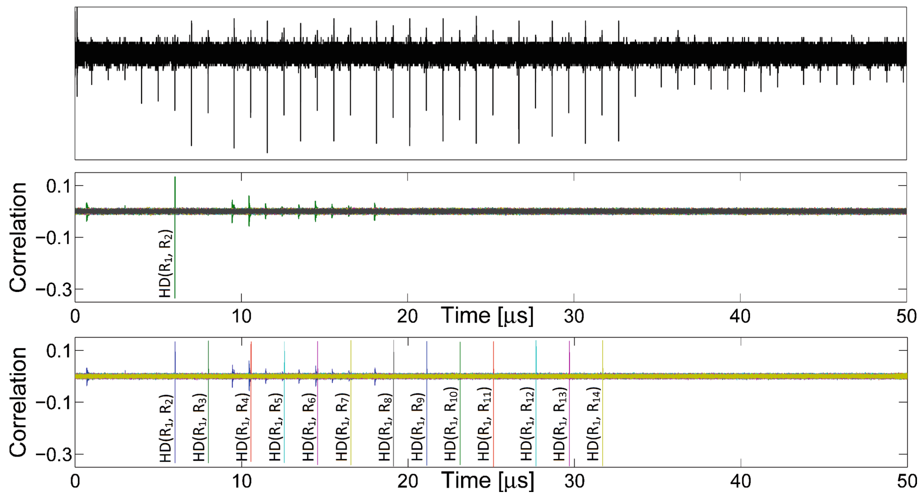


Fig. 6. Spartan-6, EM analysis, (top) a sample trace, (middle) correlation between **EM** traces and $\mathrm{HD}(R_i, R_{i+1})$ and (bottom) $\mathrm{HD}(R_1, R_{i+1}),\ 0 < i < 14$

since – as stated in Sect. 2 and shown in Fig. 1 – the decryption of the first ciphertext takes place when the second ciphertext block is sent.

3.3 Attacks

As explained in Sect. 2, the previous attack needs to search in a space of at least 2^{32}. Recalling Eq. (1), if ciphertext bytes $c^{(13)}$, $c^{(10)}$, and $c^{(7)}$ are constant we can write

$$\begin{aligned}
\Delta_{R_1,R_2}^{(0,1,2,3)} &= c^{(0,1,2,3)} \oplus k_{14}^{(0,1,2,3)} \oplus \mathrm{M1C}^{-1}\Big(\Big\langle S^{-1}\left(c^{(0)} \oplus k_{14}^{(0)}\right), S^{-1}\left(c^{(13)} \oplus k_{14}^{(13)}\right),\\
&\qquad S^{-1}\left(c^{(10)} \oplus k_{14}^{(10)}\right), S^{-1}\left(c^{(7)} \oplus k_{14}^{(7)}\right)\Big\rangle\Big) \oplus \mathrm{M1C}^{-1}\left(k_{13}^{(0,1,2,3)}\right)\\
&= \Big\langle \{\mathtt{0e}\} \bullet S^{-1}\left(c^{(0)} \oplus k_{14}^{(0)}\right) \oplus c^{(0)} \oplus \delta^{(0)},\\
&\qquad \{\mathtt{09}\} \bullet S^{-1}\left(c^{(0)} \oplus k_{14}^{(0)}\right) \oplus c^{(1)} \oplus \delta^{(1)},\\
&\qquad \{\mathtt{0d}\} \bullet S^{-1}\left(c^{(0)} \oplus k_{14}^{(0)}\right) \oplus c^{(2)} \oplus \delta^{(2)},\\
&\qquad \{\mathtt{0b}\} \bullet S^{-1}\left(c^{(0)} \oplus k_{14}^{(0)}\right) \oplus c^{(3)} \oplus \delta^{(3)}\Big\rangle\\
&= \Delta'^{(0,1,2,3)}_{R_1,R_2} \oplus \delta^{(0,1,2,3)},
\end{aligned} \tag{3}$$

where constants $\{\mathtt{0e}\}, \ldots, \{\mathtt{0b}\}$ are with respect to the MixColumns Inverse operation, and $\bullet$ the multiplication in $\mathrm{GF}(2^8)$. Further, $\delta^{(0)}, \ldots, \delta^{(3)}$ represent constants that depend on key k and ciphertext bytes 13, 10, and 7. If – in contrast to [12] – we select the ciphertexts which are given to the decryption module, and keep certain ciphertext bytes fixed (13, 10, and 7), we can perform CPA/DPA attacks by searching in a shorter spaces – as explained below – to find $k_{14}^{(0)}$.

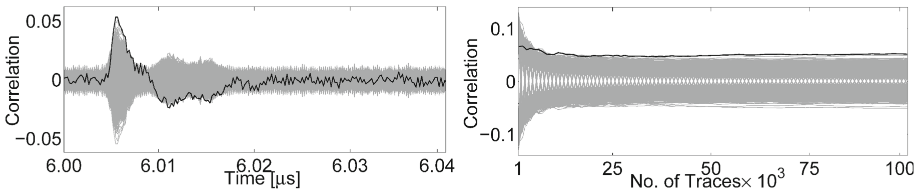


Fig. 7. Spartan-6, EM analysis, CPA in 2^{16}, HD($\Delta^{(0)}_{R_1,R_2}$) model, (a) using 100,000 traces, (b) over the number of traces

Search in a Space of 2^{16}. For example, based on Eq. (3) $\Delta^{(0)}_{R_1,R_2}$ can be predicted by guessing $k^{(0)}_{14}$ and $\delta^{(0)}$, i.e., 16 bits. Therefore, HD($\Delta^{(0)}_{R_1,R_2}$) can be used and a CPA can be performed accordingly. In this case, the disadvantage is the way that the constant $\delta^{(0)}$ contributes into the HD model. Since $\Delta^{(0)}_{R_1,R_2}$ and $\delta^{(0)}$ are linearly proportional, the use of HD model faces the *ghost peak* issue [10]. The result of such a 2^{16} attack with 100,000 traces as well as over the number of traces are shown in Fig. 7. Similarly, other bytes $\Delta^{(i\in\{1,2,3\})}_{R_1,R_2}$ can be predicted to find $k^{(0)}_{14}$ by searching in a 2^{16} space.

Search in a Space of 2^8. Similar to that of [12], the CPA/DPA attacks can be mounted targeting the bits of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$ by guessing only 8-bit $k^{(0)}_{14}$ (see Eq. (3)). Due to the 32-bit size of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$, 32 different attacks with the same target $k^{(0)}_{14}$ can be performed. Since predicting one single-bit flip out of a 128-bit register certainly leads to a low signal-to-noise ratio [10], it is favorable to combine the results of these 32 different attacks.

Heuristics. For a guessed key byte k, let us denote the result of the i-th CPA on sample point j by $\rho^{(i)}_{k,j}$. Following a similar approach to [3], we combine the results of multiple CPAs with different models by summing them up. As the constant is unknown we have to add the absolute values of the correlations as

$$\rho_{k,j} = \sum_{i=1}^{32} \left|\rho^{(i)}_{k,j}\right|$$

to combine the results of all 32 attacks. Figure 8 shows the corresponding results. It should be noted that – in contrast to their combination – none of the 32 single-bit CPA attacks could clearly distinguish the correct key byte $k^{(0)}_{14}$. Indeed, the complexity of the attack in this setting is 32×2^8.

Joint Probability. Let us suppose that the result of each CPA is a set of probabilities corresponding to the ranked key candidates. In other words, suppose that the i-th CPA on sample point j returns $p^{(i)}_{k,j}$ as the probability of the key byte k being the correct one. Since the 32 CPAs are independent of each other,

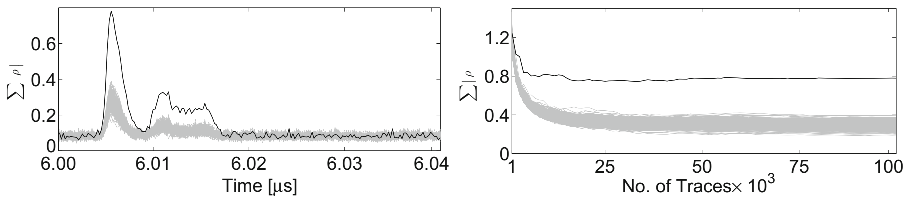


Fig. 8. Spartan-6, EM analysis, bitwise CPAs in 2^8, targeting bits of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$, combined by absolute sum, (a) using 100,000 traces, (b) over the number of traces

we can combine the results as

$$p_{k,j} = \prod_{i=1}^{32} p^{(i)}_{k,j}. \tag{4}$$

At this step, the question raised is how to project the correlation values, i.e., the result of the CPA, to probabilities? Following the concept presented in [10] and also employed in [4], we can apply Fisher's z-transform and normalize the result as

$$r^{(i)}_{k,j} = \frac{1}{2\sqrt{N-3}} \ln\left(\frac{1+\rho^{(i)}_{k,j}}{1-\rho^{(i)}_{k,j}}\right),$$

where N is the number of traces used in the CPA. Now, $r^{(i)}_{k,j}$ is a sample that can be (approximately) interpreted according to the normal distribution $\mathcal{N}(0,1)$. Therefore, we can project it to probability by

$$p^{(i)}_{k,j} = 2 \int_{-\left|r^{(i)}_{k,j}\right|}^{0} \mathsf{PDF}_{\mathcal{N}(0,1)}(t)dt \ = \ 1 - 2\mathsf{CDF}_{\mathcal{N}(0,1)}\left(-\left|r^{(i)}_{k,j}\right|\right),$$

where $\mathsf{PDF}_{\mathcal{N}(0,1)}$ and $\mathsf{CDF}_{\mathcal{N}(0,1)}$ are respectively the probability density and cumulative distribution functions of the standard normal distribution.

We have followed this procedure and calculated the joint probabilities based on Eq. (4). The corresponding results, shown in Fig. 9, indicate that this scheme is also able to combine the results of all 32 CPAs and finally reveal the key. As a side note, the probabilities can also be combined following the Bayes' theorem. However, since the Bayes' theorem results in a set of probabilities with $\sum\limits_{\forall k} p^{(i)}_{k,j} = 1$, for the sample points where none of the key candidates shows a high correlation, the probability of one key candidate (at that sample point) leads to a significantly higher value compared to that of the other candidates. This prevents us to find the most leaking sample points and distinguish the correct key. Hence, the corresponding results are omitted.

Linear Regression. From another perspective, we can map this problem to that which has been solved by means of linear regression (also known as

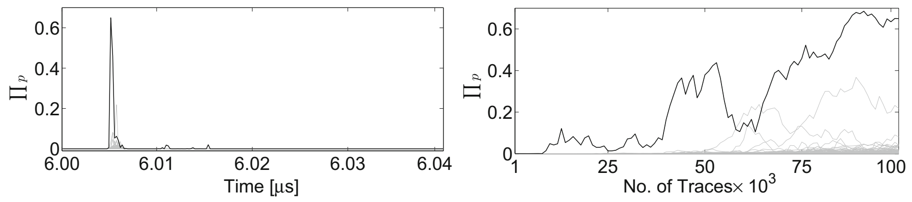


Fig. 9. Spartan-6, EM analysis, bitwise CPAs in 2^8, targeting bits of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$, combined by joint probability, (a) using 100,000 traces, (b) over the number of traces

stochastic attacks) [16]. In other words, we suppose that by guessing $k^{(0)}_{14}$ the bits of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$ contribute each with a certain weight to the leakage with respect to constants $\delta^{(0,1,2,3)}$. In more details, it is assumed that the leakage l at sample point j can be written as

$$l_j = \beta_{0,j} + \sum_{b=1}^{32} \beta_{b,j} g_b,$$

where g_b represents the b-th bit of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$.

In order to find the coefficients $\beta_{b,j} \in \mathbb{R}$ – by following the procedure of [16] – for each guessed key, we form a matrix $\mathbf{M}$ as

$$\mathbf{M} = \begin{pmatrix} 1 & g^1_1 & g^1_2 & \cdot & \cdot & \cdot & g^1_{32} \\ 1 & g^2_1 & g^2_2 & \cdot & \cdot & \cdot & g^2_{32} \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ 1 & g^N_1 & g^N_2 & \cdot & \cdot & \cdot & g^N_{32} \end{pmatrix},$$

where g^i_b represents the b-th bit of the predicted $\Delta'^{(0,1,2,3)}_{R_1,R_2}$ (based on the guessed $k^{(0)}_{14}$) for the i-th measurement (trace). As shown in [8], by means of the least square estimation, the vector of coefficients $\vec{\beta}_j = (\beta_{0,j}, \ldots, \beta_{32,j})$ is estimated as

$$\vec{\beta}_j = \underbrace{\left(\mathbf{M}^T\mathbf{M}\right)^{-1}}_{\mathbf{A}} \; \underbrace{\mathbf{M}^T\vec{l}_j}_{\vec{\alpha}_j},$$

where $\mathbf{M}^T$ stands for the transpose of the matrix $\mathbf{M}$, and $\vec{l}_j$ for the vector of leakages at sample point j (i.e., N measured traces at sample point j). $\mathbf{A}$ is a matrix of 33×33 and independent of the sample points; hence, it can be derived with processing only the associated ciphertexts. The vector $\vec{\alpha}_j$ (formed by 33 elements) is also obtained for each sample point independently. Therefore, for each guessed $k^{(0)}_{14}$, all the measured traces are processed once to derive $\mathbf{A}$ and $\vec{\alpha}_j$, $\forall j$. Consequently, $\vec{\beta}_j$, $\forall j$ are derived by $\mathbf{A}^{-1}\vec{\alpha}_j$. At the next step, instead

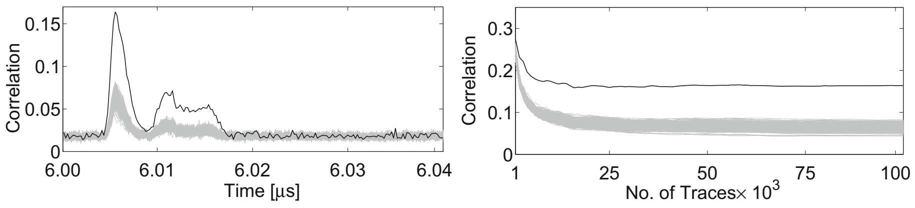


Fig. 10. Spartan-6, EM analysis, CPA in 2^8, weighted bits of $\Delta'^{(0,1,2,3)}_{R_1,R_2}$, recovered by linear regression, (a) using 100,000 traces, (b) over the number of traces

of the HD, the following model at sample point j for the i-th measurement is considered to perform a CPA:[3]

$$\hat{l}^i_j = \beta_{0,j} + \sum_{b=1}^{32} \beta_{b,j} g^i_b.$$

In other words, for each key hypothesis $k^{(0)}_{14}$, the measured traces are processed two times (first to derive the coefficients β and second to estimate the correlations). Figure 10 shows the results of this attack predicating that it outperforms all above shown attacks. Although it leads approximately to the same results as the *heuristic approach* (Fig. 8), its complexity is lower.

Other Key Bytes. The above explained procedure can be repeated for other key bytes. For example, by keeping the ciphertext bytes 0, 10, and 7 constant during the measurements, we can write

$$\begin{aligned}\Delta^{(0,1,2,3)}_{R_1,R_2} = \Big\langle & \{\texttt{0b}\} \bullet S^{-1}\left(c^{(13)} \oplus k^{(13)}_{14}\right) \oplus c^{(0)} \oplus \delta^{(0)},\\ & \{\texttt{0e}\} \bullet S^{-1}\left(c^{(13)} \oplus k^{(13)}_{14}\right) \oplus c^{(1)} \oplus \delta^{(1)},\\ & \{\texttt{09}\} \bullet S^{-1}\left(c^{(13)} \oplus k^{(13)}_{14}\right) \oplus c^{(2)} \oplus \delta^{(2)},\\ & \{\texttt{0d}\} \bullet S^{-1}\left(c^{(13)} \oplus k^{(13)}_{14}\right) \oplus c^{(3)} \oplus \delta^{(3)}\Big\rangle, \qquad (5)\end{aligned}$$

which allows the recovery of $k^{(13)}_{14}$.

It should be noted that the process of each column – in the first cipher round – is independent of the other columns. Hence, while all ciphertext bytes except 0, 4, 8, and 12 (the first row) are kept constant, four key recovery attacks (each by searching in a space of 2^8 to find the corresponding key bytes 0, 4, 8, and 12 of k_{14}) can independently be mounted. At the next step, a set of traces with fixed ciphertext bytes except the second row is measured which allows the recovery of

[3] We have also followed the suggestions of [8] to examine the squared error between the measured leakages l and estimated leakages $\hat{l}$, but our analyses showed better distinguishability when correlation is estimated instead.

key bytes 1, 5, 9, and 13. In short, we need to measure four sets of measurements, in each of which only the ciphertext bytes of one row are selected randomly, while the other 12 ciphertext bytes are kept constant (at any arbitrary value). With these four sets we are able to recover the full 128-bit last round key k_{14}.

Next Round. As the target algorithm is AES-256, we need to extend the attacks to the next decryption round. In contrast to that of [12], where $\varDelta_{R_2,R_3}$ has been considered, based on our findings (presented in Sect. 3.2) we target $\varDelta_{R_1,R_3}$, i.e., the difference between the state register at the first and the third cipher rounds.

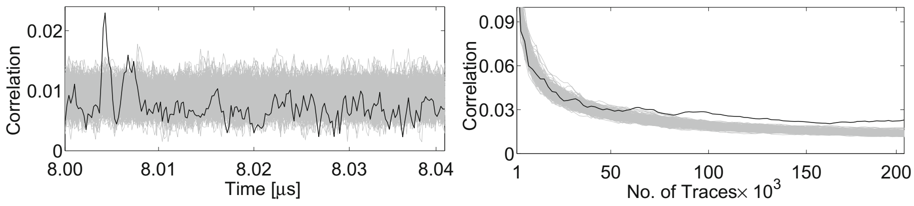


Fig. 11. Kintex-7, EM analysis, CPA in 2^8 second round, weighted bits of $\varDelta'^{(0,1,2,3)}_{R_1,R_2}$, recovered by linear regression, (a) using 200,000 traces, (b) over the number of traces

Following the same principle as explained in Sect. 2.3 (particularly Eq. (2)) we can write

$$\begin{aligned}\varDelta^{(0,1,2,3)}_{R_1,R_3} &= R^{(0,1,2,3)}_1 \oplus R^{(0,1,2,3)}_3\\ &= c^{(0,1,2,3)} \oplus k^{(0,1,2,3)}_{14} \oplus \mathrm{M1C}^{-1}\Big(\Big\langle S^{-1}\left(c'^{(0)} \oplus k'^{(0)}_{13}\right), S^{-1}\left(c'^{(13)} \oplus k'^{(13)}_{13}\right),\\ &\quad S^{-1}\left(c'^{(10)} \oplus k'^{(10)}_{13}\right), S^{-1}\left(c'^{(7)} \oplus k'^{(7)}_{13}\right)\Big\rangle\Big) \oplus \mathrm{M1C}^{-1}\left(k^{(0,1,2,3)}_{12}\right), \qquad (6)\end{aligned}$$

where $c' = \mathrm{MC}^{-1}\Big(\mathrm{SR}^{-1}\big(\mathrm{SB}^{-1}\left(c \oplus k_{14}\right)\big)\Big)$ and $k'_{13} = \mathrm{MC}^{-1}\left(k_{13}\right)$. By keeping $c'^{(13)}$, $c'^{(10)}$, and $c'^{(7)}$ constant, we can write

$$\begin{aligned}\varDelta^{(0,1,2,3)}_{R_1,R_3} = \Big\langle &\{\mathtt{0e}\} \bullet S^{-1}\left(c'^{(0)} \oplus k'^{(0)}_{13}\right) \oplus c^{(0)} \oplus \delta^{(0)},\\ &\{\mathtt{09}\} \bullet S^{-1}\left(c'^{(0)} \oplus k'^{(0)}_{13}\right) \oplus c^{(1)} \oplus \delta^{(1)},\\ &\{\mathtt{0d}\} \bullet S^{-1}\left(c'^{(0)} \oplus k'^{(0)}_{13}\right) \oplus c^{(2)} \oplus \delta^{(2)},\\ &\{\mathtt{0b}\} \bullet S^{-1}\left(c'^{(0)} \oplus k'^{(0)}_{13}\right) \oplus c^{(3)} \oplus \delta^{(3)}\Big\rangle = \varDelta'^{(0,1,2,3)}_{R_1,R_3} \oplus \delta^{(0,1,2,3)}.\end{aligned} \qquad (7)$$

Since the last round key k_{14} has been recovered, c' bytes can be arbitrary selected and the corresponding ciphertext $c = \mathrm{SB}\Big(\mathrm{SR}\big(\mathrm{MC}\left(c'\right)\big)\Big) \oplus k_{14}$ can be derived to be sent to the FPGA. Therefore, we followed the same procedure as explained for the first decryption round, and collected four sets of measurements, in each only one row of c' is selected randomly and the other bytes kept constant. This allows us to perform exactly the same attacks (each with complexity of 2^8) to find k'_{13} byte by byte to finally reveal the full 256-bit key. It is noteworthy that – in contrast to that of [12] – the attacks on the second round are as efficient as that on the first round since we are targeting Δ'_{R_1,R_3} instead of Δ'_{R_2,R_3}. As an example, the results of the attack on $k'^{(0)}_{13}$ (on the Kintex-7 device) are shown by Fig. 11.

3.4 Comparisons

Table 1 presents the results of the EM attacks on different FPGA families with different packaging. In general it can be concluded that the attacks on the devices with flip-chip technology is harder than the wire-bond ones. However, by shrinking the technology (from 65 nm to 28 nm) the attacks become harder as in case of the Artix-7 FPGA we required around 200,000 traces as one set of the measurements (with a constant row) to reveal the secrets[4], i.e., in total $2 \times 4 \times 200{,}000$ (1.6 million) traces. Since the FPGA device is in hand and control of the adversary, collecting more traces with chosen ciphertexts (if required) does not face a serious challenge. For example, with our setup we could collect each 100,000 traces of the chosen chiphertexts in around 90 min, which means that all 1.6 million traces[5] could be measured in less than a day.

Table 1. The attack performances

Family	5	6	7	
FPGA	Virtex-5	Spartan-6	Kintex-7	Artix-7
Package	Flip-chip	Wire-bond	Flip-chip	Wire-bond
Technology	65 nm	45 nm	28 nm	28 nm
Probe	RF-R50-1	RF-U5-2	RF-R50-1	RF-R50-1
Required traces for each set (row)	40,000	2,000	120,000	200,000

For the analyses, as shown in the attack results (Figs. 10 and 11) we have considered 200 sample points (either for the first or the second decryption round). These 200 sample points have been selected around the corresponding clock cycle based on the knowledge obtain from Fig. 6. We split each attack into two parts. The first part, which derives the matrix $\mathbf{A}$ and vectors $\vec{\alpha}_{j=1,\ldots,200}$, for all 2^8

[4] We realized that other components on the PCB (BASYS 3 from www.digilentinc.com) introduce noise into the EM measurements.

[5] It is done in two parts since the second part can be started when k_{14} has already been recovered.

key candidates takes 21 min using an 8-core machine @3 GHz on 100,000 traces. The results are applied in the next corresponding key-recovery CPA on the same 100,000 traces, which also takes 12 min on the same machine. In total, for a full recovery (on both rounds) using in total 1.6 million traces we require $2 \times 16 \times 2 \times (21 + 12)$ min (around 1.5 days) using the aforementioned processing unit. These numbers for sure can be decreased by more parallelization or by reducing the number of considered sample points. It should be noted that since the attacks on Spartan-6 require far less number of traces, the measurements and analyses can be done in significantly shorter time, e.g., less than an hours for the measurements and the evaluations when each set of measurements contains only 2,000 traces.

3.5 Authentication

In Virtex-6 and 7 series devices, the bitstream encryption is integrated with an on-chip bitstream keyed-Hash Message Authentication Code (HMAC). It aims at authentication of the decrypted bitstream to prove that not even a single bit was modified. Stated in [19] "Without knowledge of the AES and HMAC keys, the bitstream cannot be loaded, modified, intercepted, or cloned. HMAC provides assurance that the bitstream provided for the configuration of the FPGA was the unmodified bitstream allowed to load".

As it is also mentioned in Xilinx public documents, unlike the AES-256 key, there is no storage place for the HMAC key on the FPGA. The HMAC key is instead included in the bitstream. Our investigations revealed that the first encrypted blocks (of the encrypted bitstream) carry the HMAC key. However, since the authentication (examining the correctness of the HMAC) is performed when all bitstream blocks are transferred and decrypted (i.e., at the end of the configuration process), it does not harm the *chosen ciphertext* measurement scenario explained above. Further, after recovering the AES-256 key, the first two blocks of an original bitstream can be decrypted to derive the 256-bit HMAC key. In short, the integrated authentication scheme of all 7 series devices does not have any effect on the efficiency of our presented attack.

4 Conclusions

This work extended the known SCA attacks on the bitstream encryption feature of Xilinx. By means of a sophisticated measurement scenario, i.e., chosen ciphertext, we could reduce the search space from 2^{32} to 2^{8} for each step of the attack. This allows the attacks to be mounted by ordinary processing units, e.g., workstation PCs. We have also shown that in case of real-world attacks, the EM analysis using common ordinary EM probes are also possible, where – in contrast to all previous attacks on similar devices based on power consumption – modification of the PCB (where the FPGA is embedded) is not required. Although we have not presented the result of the attacks on Virtex-4 devices, all

FPGA families from 4, 5, 6, and 7 (where the same AES-256 decryption module is integrated) are vulnerable to the attacks presented here.

We should refer to the design and architecture of more recent Xilinx families UltraSCALE and UltraSCALE$^+$, where several security features, e.g., DPA countermeasures, have been integrated. Therefore, the attacks presented in this work are not expected to be portable to the new series devices. However, to the best of our knowledge, all 7 series devices (which are still in production) follow the same architecture and design with respect to bitstream encryption, that predicates on their susceptibility to our attacks.

Acknowledgment. The authors would like to acknowledge Alexander Jakimowic and Oliver Mischke for their help with development of the setup. The research in this work was supported in part by the DFG Research Training Group GRK 1817/1.

References

1. Balasch, J., Gierlichs, B., Verdult, R., Batina, L., Verbauwhede, I.: Power analysis of Atmel CryptoMemory – recovering keys from secure EEPROMs. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 19–34. Springer, Heidelberg (2012)
2. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
3. Doget, J., Prouff, E., Rivain, M., Standaert, F.: Univariate side channel attacks and leakage modeling. J. Crypt. Eng. **1**(2), 123–144 (2011)
4. Durvaux, F., Standaert, F.: From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. IACR Cryptology ePrint Archive, Report/536 (2015)
5. Eisenbarth, T., Kasper, T., Moradi, A., Paar, C., Salmasizadeh, M., Shalmani, M.T.M.: On the power of power analysis in the real world: a complete break of the KeeLoq code hopping scheme. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 203–220. Springer, Heidelberg (2008)
6. Heyszl, J., Mangard, S., Heinz, B., Stumpf, F., Sigl, G.: Localized electromagnetic analysis of cryptographic implementations. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 231–244. Springer, Heidelberg (2012)
7. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 388. Springer, Heidelberg (1999)
8. Lemke-Rust, K.: Models and algorithms for physical cryptanalysis. Ph.D. thesis, Ruhr University Bochum, January 2007
9. Liu, J., Yu, Y., Standaert, F.-X., Guo, Z., Gu, D., Sun, W., Ge, Y., Xie, X.: Small tweaks do not help: differential power analysis of MILENAGE implementations in 3G/4G USIM cards. In: Pernul, G., Y A Ryan, P., Weippl, E. (eds.) ESORICS. LNCS, vol. 9326, pp. 468–480. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24174-6_24
10. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks - Revealing the Secrets of Smart Cards. Springer, New York (2007)
11. Moradi, A., Barenghi, A., Kasper, T., Paar, C.: On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from Xilinx Virtex-II FPGAs. In: Computer and Communications Security, CCS, pp. 111–124. ACM (2011)

12. Moradi, A., Kasper, M., Paar, C.: Black-box side-channel attacks highlight the importance of countermeasures. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 1–18. Springer, Heidelberg (2012)
13. Moradi, A., Oswald, D., Paar, C., Swierczynski, P.: Side-channel attacks on the bitstream encryption mechanism of AlteraStratix II: facilitating black-box analysis using software reverse-engineering. In: FPGA, pp. 91–100. ACM (2013)
14. Oswald, D., Paar, C.: Breaking Mifare DESFire MF3ICD40: power analysis and templates in the real world. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 207–222. Springer, Heidelberg (2011)
15. Rao, J.R., Rohatgi, P., Scherzer, H., Tinguely, S., Attacks, P.: Or how to rapidly clone some GSM cards. In: IEEE Symposium on Security and Privacy, pp. 31–41. IEEE Computer Society (2002)
16. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
17. Swierczynski, P., Moradi, A., Oswald, D., Paar, C.: Physical security evaluation of the bitstream encryption mechanism of Altera Stratix II and Stratix III FPGAs. TRETS **7**(4), 34:1–34:23 (2015)
18. Xilinx (Kyle Wilkinson): 7 Series FPGAs Configuration User Guide (2015). http://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf
19. Xilinx (Kyle Wilkinson): Using Encryption to Secure a 7 Series FPGA Bitstream (2015). http://www.xilinx.com/support/documentation/application_notes/xapp1239-fpga-bitstream-encryption.pdf
20. Zhou, Y., Yu, Y., Standaert, F.-X., Quisquater, J.-J.: On the need of physical security for small embedded devices: a case study with COMP128-1 implementations in SIM cards. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 230–238. Springer, Heidelberg (2013)