Read The Free Bitstream

– Extracting Secrets from Protected Hardware –

Amir Moradi
Horst Görtz Institute
for IT-Security
Ruhr-University Bochum,
Germany
amir.moradi@rub.de

Alessandro Barenghi
Dipartimento di
Elettronica e Informazione
Politecnico di Milano, Italy
barenghi@elet.polimi.it

Timo Kasper
Horst Görtz Institute
for IT-Security
Ruhr-University Bochum,
Germany
timo.kasper@rub.de

Christof Paar
Horst Görtz Institute
for IT-Security
Ruhr-University Bochum,
Germany
christof.paar@rub.de

Abstract

Over the last two decades FPGAs have become central components for many advanced digital systems, e.g., video signal processing, network routers, data acquisition and military systems. In order to protect the intellectual property and to prevent fraud, e.g., by cloning an FPGA or manipulating its content, many current FPGAs employ a bitstream encryption feature. We develop a successful attack on the bitstream encryption engine integrated in the widespread Virtex-II Pro FPGAs from Xilinx, using side-channel analysis. After measuring the power consumption of a single power-up of the device and a modest amount of off-line computation, we are able to recover all three different keys used by its triple DES module. Our method allows extracting secret keys from any real-world device where the bitstream encryption feature of Virtex-II Pro is enabled. As a consequence, the target product can be cloned and manipulated at will of the attacker. Also, more advanced attacks such as reverse engineering or the introduction of hardware Trojans become potential threats. As part of the side-channel attack, we were able to deduce certain internals of the hardware encryption engine. To our knowledge, this is the first attack against the bitstream encryption of a commercial FPGA reported in the open literature.

1. INTRODUCTION

Electronic devices have become essential parts of our private life (e.g., cell phones, e-readers or set-top boxes), at work (e.g., laptops or routers) and in industry (e.g., controls systems or sensors). Virtually all of today's IT, communication and consumer electronic systems employ digital technology. As a flip side of this well-known development, reverse-engineering and product piracy has become an important issue for a wide variety of electronic products since the creation of exact copies of digital information is often straightforward. Examples include consumer products, network routers and set-top boxes, or military systems. The threat is not limited to merely counterfeiting products; once a product has been reverse-engineered it becomes vulnerable to various other attacks. For instance, ill-intended malfunctioning of the device or circumventing business models based on the electronic content, which is regularly happening in the pay-TV sector, become possible. Not only consumer products, such as hard disk malware [2], but also industrial and military applications can be affected. For instance, the recent discovery of details of the STUXNET virus shows the potential implications that embedded malware can have [12]. Another flavor of malicious manipulation of digital systems was described in a 2005 report by the US Defense Science Board, where the clandestine introduction of hardware Trojans was underlined as a serious threat [1]. In summary, protection of digital secrets and intellectual property is a key factor for developing and successfully marketing electronic products nowadays.

1.1 FPGA Basics

When developing embedded systems, the main choices are software, e.g., in the form of embedded microcrocessors, or hardware, e.g., in the form of application-specific integrated circuits (ASICs). A third form of device, field programmable gate arrays (FPGAs), combine some advantages of software (fast development, low non-recurring engineering costs) with those of hardware (performance, relative power efficiency). These advantages have made FPGAs an important fixture in embedded system design, especially for applications that require heavy processing, e.g., for routing, signal processing or encryption. Modern high-end FPGAs have the functional equivalent of several tens of millions of Boolean gates, making them formidable devices for a large spectrum of applications. Most of today's FPGAs are (re)configured with bitstreams which completely determine the functionality of the device. As described in Sect. 2.1, programming them is quite similar to developing software for a microprocessor, allowing for fast development and a quick time-to-market.

Nowadays FPGAs have applications in banking, defense, aerospace, and many sophisticated commercial technical applications such as video signal processing, e.g., for HDTV, or network routing. Even hitherto totally mechanical devices such as guns nowadays incorporate FPGAs, e.g., the XM25 Individual Airburst Weapon System [9]. Noteworthy for this contribution, satellite communication and other mission-critical systems as well as high-security applications also employ FPGAs [20]. As a key advantage, a product relying on FPGAs can be regularly improved during its life-cycle by simply changing the bitstream, e.g., if a design bug is found or to provide new functionality. To name two examples, today's network routers are easily upgraded in the field via the Internet and set-top boxes obtain an updated firmware via cable or a satellite link. One of the disadvantages of FPGAs, especially with respect to custom hardware such as ASICs, is that an attacker who has access to the bitstream can clone the system and extract the intellectual property of the design. Note that the bitstream is in the vast majority of systems stored externally to the FPGA in a dedicated configuration memory and is from there loaded into the FPGA on every power-up or reset — an adversary wire-tapping the relevant data signals can hence easily monitor the bitstream. The main answer of the industry for protecting the design is a security feature called *bitstream encryption.*

1.2 Bitstream Encryption Basics

The idea of bitstream encryption is to establish end-to-end confidentiality by means of symmetric cryptography. It protects the entire path of an FPGA bitstream: the development environment of the manufacturer, the insecure channel into the product, the storage inside the product, and finally the loading of the design into the FPGA. Figure 1 illustrates an example for a secure firmware upgrade of an FPGA-based network router. As a prerequisite, the manufacturer and the FPGA in the network router possess the same secret key, that should be created individually for every device and securely stored both inside the FPGA (k_{FPGA}) and at the manufacturer site (k_{design}). After generating the bitstream, the designer encrypts it with a secure symmetric cipher such as triple DES or AES using a secret key k_{design}. This encrypted bitstream can now be safely sent, e.g., via the Internet, to the configuration memory of the target network router, from where it is loaded into the FPGA. The latter possesses an internal decryption engine and uses its secret key k_{FPGA} to decrypt each the bitstream. Finally, the FPGA configures its internal circuitry according to the decrypted bitstream. The configuration is successful if and only if the secret keys used for the encryption and decryption of the bitstream are identical, i.e., $k_{design} = k_{FPGA}$.

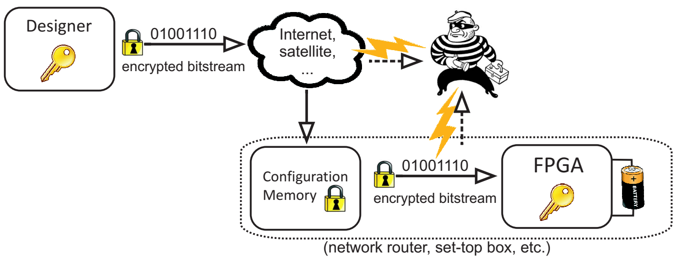


Figure 1: Bitstream encryption enables to securely transfer the content of an FPGA, e.g., a firmware update for a network router via the Internet.

A third party who gets hold of the encrypted bitstream — e.g., from the Internet or by wire-tapping the internal data bus that is used for the configuration inside the product — will not be able to extract any useful information. Without knowing the secret key for the decryption, she will not be able to deduce the design or configure another FPGA correctly. As a major consequence, counterfeiting or attacking the product becomes infeasible, and no confidential or proprietary information contained in the bitstream can fall into the hands of the attacker.

1.3 Side-Channel Attacks

Side-channel attacks exploit physical information leakage of a cryptographic implementation in order to extract secret information, in particular the cryptographic key. In the case of power analysis, the consumption of 7electric current or the electromagnetic emanations of the cryptographic device are used as a side channel for key extraction. The underlying principle is a divide-and-conquer approach, i.e., small parts of the key, e.g., 6 bits, are guessed and the hypotheses are verified. This process is repeated until the whole key is revealed. Contrary to mathematical or brute-force attacks, where typically pairs of plaintexts and ciphertexts are required for a key-recovery, side-channel attacks require only one of them, i.e., either the plaintext or the ciphertext [15]. While the open literature reports a number of successful attacks against cryptographic architectures implemented on the FPGA *fabric* [25, 27], no publicly known attack on the bitstream encryption has been reported in the open literature. We also refer to a number of surveys dealing with FPGA security which have not addressed bitstream reverse engineering as a real-world threat [30, 14].

1.4 Content of this Paper

In this paper we investigate the level of security provided by the bitstream encryption used in the widespread Virtex-II family of FPGAs produced by Xilinx. A detailed description of these FPGAs and their bitstream encryption feature, as well as an introduction to power analysis, are given in Sect. 2. The manufacturer claims that the bitstream encryption as implemented in Xilinx Virtex-II Pro FPGAs can thwart even most Class III attacks [28], i.e., attacks by well-funded intelligent agencies [6]. However, our research does not support this claim as shown in Sect. 3: starting without any previous knowledge about the implementation, i.e., a typical black box scenario, we demonstrate step-by-step how to conclude from the power consumption and timing behaviour of the FPGA to the detailed internal structure of the decryption hardware. Finally, we develop a side-channel attack that exploits the power consumption of the FPGA during the decryption of one bitstream to recover the secret keys used for the bitstream encryption. Some of the dramatic implications of our attack are illustrated in Sect. 4.

2. PRELIMINARIES

This section introduces Field Programmable Gate Arrays (FPGA) in general, and in particular the Virtex-II Pro device and its security features. We further describe the side-channel analysis and digital signal processing techniques used for our attack.

2.1 Flexible Hardware: FPGA Details

An FPGA is a reconfigurable integrated circuit that can host highly complex digital circuitry, e.g., a complete microcontroller, digital signal processing algorithms, or almost any other design can be put in silicon.

Analyzing the worldwide **FPGA market**, *Xilinx* (51.2 %) and *Altera* (35.5 %) together account for more than 85 % market share, followed by *Actel* and *Lattice* with about 6 % market share each (Gartner Inc., 2008) [13]. Both market leaders offer security solutions for commercial and military environments that aim at protecting the bitstream by means of symmetric encryption. Altera programs the respective secret keys in non-volatile e-fuses on the silicon die of the FPGA that are covered by layers of metal to hinder extraction of the key by invasive physical attacks [8]. Xilinx on the other hand states that programmable fuses are easy to reverse-engineer [20] and instead promotes a battery-backed solution to store the keys in volatile memory, ensuring that the key is instantly zeroed if the FPGA is removed from the product, e.g., to conduct physical attacks. In this paper we focus on the advertised to be more secure solution of the — in terms of FPGAs sold — leading manufacturer Xilinx.

The essential **building blocks** of an FPGA are configurable logical blocks (CLB) which consist of *slices.* Each slice in turn contains small look-up tables (LUTs) that realize arbitrary combinational functions, i.e., small Boolean circuits, and usually also simple memory elements, e.g., latches storing one bit. Programmable switches allow altering of the signal path inside the slices, as well as routing of the data to other CLBs on the FPGA. Depending on the particular type of FPGA various other auxiliary resources are available, for instance, multiply-and-add units, BlockRAMs that can be accessed by several CLBs, and even entire CPUs. By appropriately configuring the slices and combining the inputs and outputs of tens of thousands CLBs, an FPGA can realize the most elaborate tasks in parallel and hence achieves very high computational power. Obviously, the vast complexity of this configuration process cannot be managed manually, thus software tools are required for the design of an application. The tools ultimately generate the configuration information, i.e., the bitstream, for the FPGA.

The **design flow** for reconfigurable hardware starts with expressing the function to be realized by means of a high-level hardware description language (HDL) such as VHDL or Verilog. The FPGA manufacturers provide integrated development environments for their products, e.g., the Xilinx ISE Design Suite [4]. They allow HDL descriptions to be synthesized into a schematic representing the internal wiring of the corresponding FPGA, commonly called *netlist*. The synthesized netlist can be simulated on the development PC for debugging the implementation. Finally, the netlist is translated into a low-level circuit description, the *bitstream*, that specifies the exact content of each slice, each programmable switch, and every other configurable component inside the FPGA. Configuring the target device with this bitstream uniquely sets the initial state of the whole circuitry to realize the designed function.

2.2 Device Under Attack: Virtex-II Pro

As the target for our analysis we opted for a Virtex-II Pro XC2VP7 FPGA. According to the data sheet [33] its silicon die consists of nine metal layers manufactured in a 130 nm process with 90 nm high-speed transistors, allowing for designs with clock frequencies up to 600 MHz for internal components. Amongst others, the FPGA provides more than $11,000$ internal registers and LUTs and an embedded PowerPC 401 processor core, wired to the fabric, in case a general purpose processor is needed alongside application specific hardware. For configuring all of its internal circuitry the FPGA requires a bitstream consisting of $560,700$ bytes.

During our analysis, the power supply pins of the FPGA are of special interest, as they form the entry point for the side channel. The pins are divided in three main groups according to the parts of the device they are providing energy to: VCC_{INT} for a 1.5 V supply of the internal core logic, VCC_{AUX} for a 3.3 V supply of auxiliary circuits, and VCC_{O} for a 3.3 V supply of the input and output buffers. Note, that the 1.5 V of the core voltage VCC_{INT} can be distinguished from the other 3.3 V supply voltages and can hence be easily identified on the unknown printed circuit board of a real-world product.

Both Xilinx and Altera FPGAs rely on volatile static random access memory (SRAM) for holding the configuration information and, as a consequence, loose the latter when the power is switched off. In order to keep the crucial cryptographic key, in Xilinx products a lithium battery providing 1.0–3.6 V has to be connected to the VCC_{Batt} pin. The battery supplies the SRAM-based secret key storage used for the bitstream decryption and provides the basis to use Virtex-II Pro devices in FIPS 140-2 Level 4 security devices, the highest security standard indicated by NIST. The standard mandates that all cryptographic keys must be instantly zeroed if physical tamper actions are detected on the secure device. In case the battery is disconnected and the FPGA is not powered up the keys are instantly flushed away, forcing the customer to return the device to the producer for re-keying. While power is applied to the VCC_{AUX} pin of a Virtex-II Pro, the battery is not required to buffer the key storage and can for instance be replaced [31].

2.2.1 *Configuration Process and Protocols*

As stated above, the configuration of the FPGA is completely erased every time the FPGA is powered off. It is thus necessary to reprogram the device after every boot, even when it is deployed in the field. Xilinx provides five different configuration protocols to configure the device, namely Master or Slave Serial mode, Master or Slave SelectMAP mode and IEEE-1149.1-2001 standard JTAG boundary scanning mode [5]. The first two modes input the bitstream into the device, one bit per clock cycle, through a specific pin which serves as a data-in line. The SelectMAP modes represent an evolution of the simple serial mode as they are able to send a byte of the bitstream at a time, thus decreasing significantly the time needed for configuration. The JTAG mode allows to input the whole bitstream into the device considering the internal configuration memory as one of the data registers of a standard JTAG chain. It is thus possible to insert the configuration information one bit at a time via JTAG, taking proper care of setting all the other devices on the JTAG chain into bypass mode.

Due to the widespread adoption and standardization of the JTAG protocol, we chose the JTAG interface of the FPGA for supplying the bitstream during our attack. The vast majority of commercial products retain the JTAG line contacts for post-production testing, hence this port is also the most suitable for real-world attacks. The JTAG interface is based on four lines: two for data input and output (TDI and TDO), one to supply the clock to the whole JTAG chain (TCK) and one to supply commands to the finite state machine implementing the JTAG port on every device (TMS). All the modules belonging to a JTAG chain, e.g., the configuration memory and the FPGA itself, have their TDI and TDO port daisy chained together and form a single loop. The number of devices connected to the JTAG testing chain and the order in which they are linked to the JTAG loop can be easily reverse-engineered [26].

2.2.2 *Bitstream Encryption in the Virtex 2*

The bitstream encryption feature [19] enables to configure the Virtex-II Pro FPGA with a bitstream that is encrypted by means of the symmetric-key cipher triple DES [32]. *Triple DES* is a block cipher constructed by chaining three subsequent executions of the Data Encryption Standard [23] (DES) by using either two or three different DES keys. In order to provide backwards compatibility, the three executions of DES are actually an encryption-decryption-encryption pattern: in this way, if a single key is employed for all three cipher instances, the triple DES is effectively reduced to a single DES. The DES cipher encrypts blocks of 64 bits employing a 56-bit key, hence the possible key lengths of triple DES are 112 or 168 bits. Whilst single DES is vulnerable to brute-force attacks due to its short key length, e.g., using the COPACOBANA code breaking machine [17] or with commodity hardware such as GPUs [7], there are no theoretical or brute-force attacks against triple DES known with any realistic chance of success. Hence, neither mathematical cryptanalysis nor an exhaustive search can endanger the secrecy of a design protected by the bitstream encryption.

Since the bitstream is processed by the cipher in blocks of 64 bits, *Cipher Block Chaining* (CBC) [22] is implemented to combine subsequent blocks of the bitstream. During a decryption, this mode of operation defines that the decrypted plaintext of one block should be added via a bitwise XOR to the next ciphertext in order to form the input block for the next decryption. Since the first block to be decrypted does not have a predecessor, a 64-bit initialization vector (IV) is used to mask it. This IV may be publicly known and can hence be transmitted in plain without loss of security. The analyzed FPGA can store up to six single DES encryption keys, which can be independently marked for use in either the first, the second or the third DES run of the triple DES. This feature allows for either 3 key sets (for 2-key standard triple DES) or 2 key sets (for use with 3-key triple DES) and enables the manufacturer to update the bitstream even if a key set has been compromised. Xilinx states that there is no read port for the key storage, except for the one internally employed by the decryption engine [32]. The only way to program the keys into the device is through the JTAG programming mode, although no details of the key entering mode are provided by Xilinx in the technical specification. In order to avoid incorrect configuration of the FPGA due to a faulty decrypted bitstream and the subsequent possible damage to the device, Xilinx states that the FPGA performs a CRC check on the decrypted bitstream. For this purpose, a CRC16 standard checksum embedded in the configuration file is being used.

2.3 Power Analysis and Power Models

The power consumption of modern CMOS-based devices mostly depends on the ongoing switching activity. It is possible to exploit this relation in order to gain insights into the values being involved in a computation. For power analysis, the power dissipation is measured during the regular functioning of a secure device in order to infer the secret keys. The literature divides this class of attacks into two general families, "simple" (SPA) and "differential" (DPA) attacks [18]. *SPA attacks* involve visually interpreting power consumption measurements as a function of time in order to detect data-dependent properties between the computed value and the power consumption, in precise intervals of time. This methodology assumes that it is possible to discern a subset of keys thanks to their peculiar consumption behavior, e.g., to distinguish a square and multiply step in a modular exponentiation from a simple squaring. In public-key algorithms such as RSA this can lead to a complete leakage of the secret key. SPA is rarely applicable against symmetric ciphers, as used in our target device. *DPA attacks* rely on building a power model of a computing circuit, employing the input values to the circuit as inputs to the model, and a set of possible values of the unknown key as a parameter. To verify a correct key hypothesis, the attacker correlates the predicted values with the actual measurements from the device under attack through a statistical tool of her choice. Commonly, for a DPA it is assumed that the values of the measurement set and the hypothesis set are normally distributed random variables with mean μ and standard deviation σ, where μ is the mean consumption of the circuit for a precise key value at a specific time instant. Since the dynamic power consumption of a device is caused by the switching activity of the logic gates [16], a proper dynamic consumption model tries to express the power consumption depending on the intensity of the switching activity. This switching activity may be caused, e.g., by the combinational logic computing the result of a Boolean function, or by a latch storing a single bit value. A first order approximation of the switching activity of many circuits is provided by the *Hamming weight* (HW) of the input value. This results in a quite approximative model and has the advantage of requiring no precise knowledge about the implementation, since only the data processed at one instant in time has to be predicted. The switching activity induced by a latch storing a value is often better modeled by the *Hamming distance* (HD) between the former value stored by the latch and the new one. A latch toggles internally (and thereby consumes noticable amounts of power) only if the previous value being held is different from the new one to be memorized, i.e., when their HD is 1 [11]. Consequentially, the model requires to predict two intermediate values, i.e., those stored in a register before and after the targeted operation. This requires some additional knowledge about the implementation, compared to a HW model.

Once a power model has been chosen, the attacker predicts an intermediate value depending on both a known input value and a part of the secret key. She then computes a set of hypothetical power consumption values, one for each of the possible values taken by the part of the secret key. The size of the part of the secret key hypothesized during one step of the attack is a trade-off between building a faithful model of the power consumption, which requires more key bits to be considered in a single time, and the computational complexity involved in computing a power hypothesis for every possible value which may be taken by the key portion. However, since attacking one key portion is performed independently from attacking the others, the attacker is free to divide the whole key in portions that are small enough to be processed. To recover the whole key, multiple attacks are performed (on the same set of measurements). After collecting a large number of measurements during the cryptographic operation to be attacked, the adversary employs a statistical tool to compare the hypothesized power values with those that have been measured, in order to infer the correct value of the key part. The typical methodology to correlate the predicted and actual power consumption of the circuit is to employ Pearson's linear correlation coefficient: if the model correctly predicts the consumption of the circuit, the linear correlation among the synthetic values and the recorded ones will be close to 1, while it is expected for a wrong model to have negligible correlation values. Conducting the correlation analysis time-wise, i.e., for each time instance of the consumption values recorded from the circuit, gives the attacker additional information about the exact moment when a side-channel leakage occurs in the power traces: the values of the correlation coefficient of the correct key hypothesis spike only in the time interval in which the operation is executed.

2.3.1 Processing the Measurements

For a successful power analysis attack the collected power traces need to fulfill a number of requirements. The first and foremost, due to the time-wise nature of the analysis, is that all the collected measurements should share the same starting point in time, i.e., they should be **well aligned**. Due to either instrumental issues or the fact that the instant of the beginning of a sensitive operation is not always in full control of the attacker, it is necessary to process the obtained traces in order to compensate for the possible phase shifts. The problem of realignment can be solved through choosing a single trace as a reference and detecting the most likely time delay that needs to be applied to each other measurement in order to achieve the best match between the traces. Since the measurements are taken in very comparable situations (i.e., encryptions of different plaintexts with the same algorithm), the most natural figure of merit to detect the appropriate time delays is cross-correlation. Cross-correlation is a measure of the similarity of two waveforms as a function of a time-lag applied to one of them. The time delay maximizing the cross-correlation between the reference and the current trace is taken as the optimal time delay to achieve alignment.

A further key point for a successful power analysis is minimizing the amount of noise in the traces. Regardless of the quality of the measurement setup the measurements always contain noise, since the traces represent an aggregate measurement of the power consumption of the entire chip and its environment. **Digital filtering** techniques can often be employed to separate the side-channel leakage from the unwanted signals and thereby extract only the relevant information. Their usage turns out to be of crucial importance when dealing with large digital components as in [10].

3. ATTACKING A BLACK BOX

This section will provide a detailed description of the steps performed to successfully reverse engineer and break the protection scheme of Xilinx's Virtex-II Pro. The first subsection illustrates the implementation details of the encrypted bitstream concept that were gathered through the analysis of the output of the Xilinx ISE synthesis tool. After presenting the in-vitro analysis on the bitstream, the section will de-

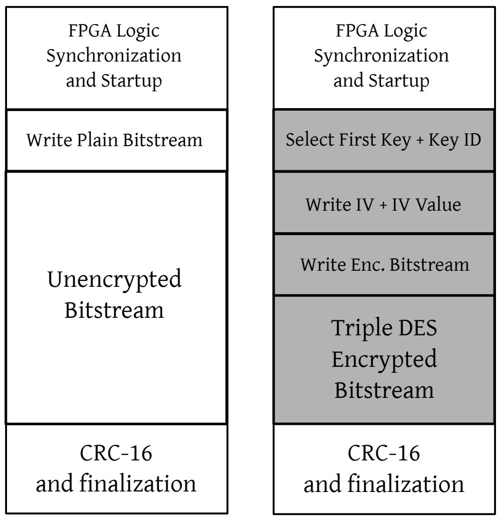

FPGA Logic Synchronization and Startup
Write Plain Bitstream
Unencrypted Bitstream
CRC-16 and finalization
FPGA Logic Synchronization and Startup
Select First Key + Key ID
Write IV + IV Value
Write Enc. Bitstream
Triple DES Encrypted Bitstream
CRC-16 and finalization

Figure 2: Comparison between the structure of an encrypted and an unencrypted bitstream

scribe the custom communication device we built to program the FPGA and the measurement setup employed to record the power consumption profiles of the Virtex-II Pro. Subsequently we describe the preliminary chip analysis oriented at identifying the moment in time when the decryption is performed and provide an analysis of the power profile of the device. The section then scrutinizes the inner architecture of the DES decryption engine according to our findings about the time instances in which the intermediate values are processed. Finally, the precise attack techniques that allow to recover the whole triple DES key used for the decryption of the bitstream are illustrated and the minimal number of traces required for the full-key recovery is determined.

3.1 Reverse Engineering of the Bitstream

As a first step the precise format of the bitstream needs to be analyzed. Understanding the bitstream is a prerequisite to find out how the configuration information is processed by the internal decryption engine of the device. We assume that the attacker is able to monitor the entire encrypted bitstream of the target FPGA, since the bitstream of an FPGA retains the same size regardless of the design implemented(the FPGA needs to be configured in full at least during its first boot). Moreover, in case the bitstream encryption feature is enabled, the device cannot use the partial reconfiguration features — it is hence forced to perform a full reconfiguration for each update. To obtain the full (encrypted) bitstream an attacker has two options: *i*) she can wire-tap the JTAG data and command lines in order to eavesdrop on the communication while the FPGA is being configured, or *ii*) she can read out the content of the non-volatile memory attached to the FPGA through the same JTAG chain that is used for the configuration. From now on, we assume that the attacker is in possession of the full encrypted bitstream through one of these methods.

Comparing Bitstreams.

While the Virtex-II Pro user guide [31] from Xilinx documents all the common configuration registers and gives a brief description of the inner addressing modes of the device, the internal registers driving the decryption are not explicitly mentioned.

In order to understand which parts of the bitstream drive the decryption, we start the reverse engineering on the basis of a very simple test design comprising just a single boolean gate. We synthesized the test design once without activating the bitstream encryption feature, to obtain an unencrypted bitstream, and then again with varying keys for the encryption, producing several bitstreams that are encrypted with the different keys. In the following we reveal the relevant details of the configuration by comparing the files containing the different bitstreams.

The differences deduced from the comparison are summarized in Fig. 2. The content of the bitstream is organized in packets, preceded by a single 32-bit synchronization word (namely, `0xAA995566`). This initialization header is the same for both encrypted and unencrypted bitstreams and has the purpose to synchronize the FPGA logic and reset the inner programming logic to a default state. After this phase, the unencrypted variant of the bitstream simply consists of a large packet with the whole configuration information. In contrast, the encrypted bitstream contains some extra information for initializing the cryptographic engine. The first value indexes the first key in the key storage to be employed for the triple DES decryption. The two remaining keys are determined by flags marking two cells of the 6-keys buffer as "middle" and "last". After the key indexes, the value of the IV required for the CBC mode is passed in plain. Following the IV value, the bitstream continues with a large packet containing the whole configuration information encrypted with triple DES. The writing command is the same as for the unencrypted bitstream, with the exception of an extra bit which is set to 1: we thus infer that setting the bit effectively enables the decryption engine. The next step was to reverse-engineer how the 64-bit secret keys input to the ISE synthesis tool are processed before being used for the encryption of a bitstream. In contrast to the DES standard, which suggests either to discard the eighth bit of every byte of the DES key or to employ it for a parity check, we have discovered that the ISE tool discards the full first byte of the 64-bit supplied key in order to obtain the 56-bit key for the DES encryption/decryption. As a final check to all our inferences and in order to understand the endianness of the bitstream, we generated a special bitstream that is encrypted with triple DES employing one of the known *weak keys* of DES — the same for all three executions of DES. These special weak keys have the property that a DES encryption involving one of them becomes an involutary operation, i.e., encrypting a second time with the same weak key is identical to a decryption and hence yields the plaintext. Through generating a bitstream that is encrypted with weak keys and with the IV of the CBC mode set to zero, due to the properties of the CBC mode the second 64-bit block contained in the bitstream is equal to the plaintext corresponding to the first block in the bitstream — thus we obtain a correctly deciphered plaintext in the generated bitstream file. Comparing this deciphered plaintext with the available unencrypted bitstream we confirmed that we correctly understood the triple DES engine input format and in particular the order in which the bits are processed internally.

Timing Issues.

The last information we were able to infer before tackling the device pertains the minimum throughput of the internal decryption engine. The Xilinx user guide reports that the maximum clock rate to be externally supplied during JTAG programming of an encrypted bitstream is $f_{clk} = 33\,\text{MHz}$. In every clock cycle, one new bit of the bitstream is supplied to the FPGA. This in turn implies that the triple DES engine must be able to perform a full triple DES decryption of a 64-bit block in a time less than $64 \times \text{T}_{clk}$, where $\text{T}_{clk} = \frac{1}{f_{clk}}$ denotes the duration of a clock cycle. This gives us the expected time for a full triple DES of around $64 \times \text{T}_{clk} = 1.94\,\mu\text{s}$ which implies that, assuming all 48 rounds of the triple DES take roughly the same time, a single round of the DES must be executed in around 40 ns. Accordingly, the hypothesis of the DES being perfomed by means of a software implementation running on an internal microcontroller becomes highly unlikely, since it would be too slow — instead we now assume a hardware implementation. If the decryption was realized by implementing one DES round in hardware and execute it 48 times for the triple DES, the above details about the timing give a lower bound for the clock rate of the inner cryptographic engine of 24.75 MHz.

3.2 Customizing the Measurement Setup

After analyzing the format of the bitstream and the specifications of the target device, we moved on to develop a measurement workbench in order to record the power profile of our target FPGA during its operation in a real-world scenario. The first step in this direction was to develop a customized *communication module* that is able to correctly configure an FPGA via JTAG. Hence, we designed an in-system programmable board that is based on an Atmel ATMega256 8-bit microcontroller and provides a JTAG port, a universal serial bus (USB) as well as a dedicated pin for triggering the oscilloscope. We implemented a framework on the microcontroller comprising the JTAG protocol and a serial protocol for communicating via USB. The firmware allows to freeze the configuration process through stopping the clock signal fed to the Virtex-II Pro and enables to issue the trigger signal to the oscilloscope before the clocking in of any chosen bit, i.e., with a resolution of 125 ns. While the fixed header of the bitstream (see Fig. 2) fits into the memory of the microcontroller, the remaining bits to be sent are provided by the control PC and sent to the communication module. The latter then wraps the bitstream in the JTAG protocol and forwards it to the DUT while issuing trigger signals at the appropriate time instants.

As the *target platform* for attacking the bitstream encryption we chose the customized FPGA development board SASEBO [3]. It contains an XC2VP7 Virtex-II Pro FPGA and provides stable and suitable voltages by means of onboard voltage regulators. A JTAG connector is provided to configure both the FPGA and a dedicated PROM which are connected in a daisy chain form. During the analyses we slightly modified the board by inserting resistors that allow measuring the power consumption of the VCC_{INT}, VCC_{AUX} and *GND* paths. Similar modifications are also required for attacking other real-world products comprising a bitstream encryption.

A LeCroy WP715Zi 1.5 GHz digital oscilloscope with a maximum sampling rate of 20 GS/s serves to measure and capture the instantaneous power consumption of the target FPGA at a maximum vertical precision of 2 mV/division. The program feeding the encrypted bitstreams to our customized programmer and controlling the acquisition of the power consumption traces runs on the oscilloscope itself, i.e., the communication module is connected to it via USB. The employed probe connecting the oscilloscope to the measurement resistor on the board is a LeCroy AP033 active differential probe, which includes a low noise 10x analog amplifier that boosts the effective vertical resolution of our measurement subsystem to 200 μV/division.

For verifying that our setup allows to correctly configure the target FPGA with custom (encrypted) bitstreams, we again synthesized our test design comprising a single boolean gate and connected the appropriate pins of the FPGA to two external switches serving as inputs and an LED displaying the output of the logical operation. Experimenting with various self-generated bitstreams, e.g., by means of arbitrary IVs, we used the simple circuit as a practical means to debug the proper configuration of the FPGA and successfully verified the functionality of our setup.

3.3 Timing and Power Profile Analysis

With the framework for configuring the FPGA and acquiring its power consumption at hand we now proceed to analyze the power profile of the target FPGA, in order to identify the point in time when the targeted triple DES decryption takes place. We started our analyses from the VCC_{INT} line, which, according to the specifications provided by Xilinx, powers the whole FPGA fabric and inner circuits. This line turned out to be the actual line feeding the decryption engine. For the sake of completeness, the detection analyses have been repeated also on VCC_{AUX}, which did not show any change in the power consumption whether or not the decryption engine is enabled. We may thus conclude that only VCC_{INT} is effectively powering the FPGA. To record the power consumption of the internal circuits connected to VCC_{INT}, there are two options: the power consumption can be sampled either by measuring the voltage drop across resistor between VCC_{INT} and the device, or one between the device and GND.

Trying the second option first, we detected a strong echo from the ground plane, implying a strong variation of the offset of the measured power consumption that rendered the power traces useless for further analysis. The disturbing effect turned out to be related to the activities at the input and output ports of the FPGA during the configuration, i.e., the communication via JTAG: the power consumption of all voltage supply inputs (VCC_{INT}, VCC_{AUX} and VCC_O) is summed up when measuring at the GND path, as a consequence the data and clock signal of the JTAG port caused the undesired variations in the offset. As a remedy for the problem we finally decided to acquire all our measurements between VCC_{INT} and the device and thereby solely record the power consumption of this pin, separated from all other activities on the other power pins.

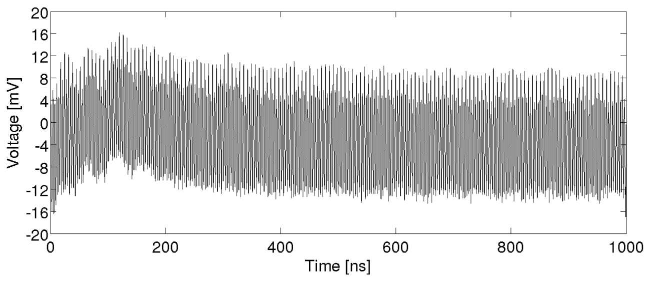
For learning more about the internal configuration process we performed tests with bitstreams that were generated intentionally wrong, i.e., configuring an FPGA with them results in an unpredictable behavior. During the tests, the consumption of the FPGA spiked after a while, the device heated up and the configuration process effectively stopped: we proceeded no further in order to avoid damage to the device. These findings let us suggest that the configuration information is written to the SRAM memory of the configuration fabric instantly upon reception of each configuration block, even before the actual boot command is given and before the CRC checksum is properly verified.

The first step in the detection of the time period in which the decryption takes place was done through recording power traces during the loading of 64 bits of the bitstream. Two long traces have been collected: one from an encrypted bitstream and the other one from a plain bitstream. Through comparison we spotted the key difference in power consumption after the second bit of the 64-bit block is clocked in. Fig. 3 depicts the power consumption of the FPGA for the case of an encrypted bitstream (at the bottom) and its unencrypted counterpart (at the top), revealing a clear increment in the dynamic power consumption occurring at this time instant only if the bitstream encryption feature is used.

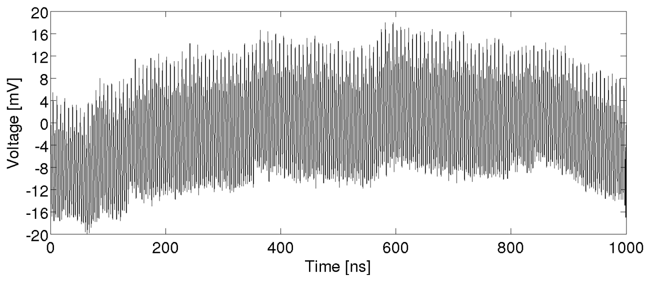
Filtering the Measurements.

A strong oscillating signal with a frequency of 295 MHz can be noticed in both traces. Being present also in the unencrypted bitstream trace we followed the intuition that it is not related to the decryption engine and tried to remove it by means of a band block filter. The latter is realized in the digital domain and suppresses the components of the signal at 295 MHz, employing a narrow Chebyshev type 2 window in order to avoid any disturbances of the other harmonic components. The results of the filtering are depicted in Fig. 4: it is now possible to clearly distinguish the shape of the three DES executions, followed by the same peak in power consumption regarding what we suppose is the write-back operation of the 64-bit word into the configuration fabric.

In order to confirm that we did not discard any information relevant for further power analysis, we computed the variance of both the filtered measurements and also the part

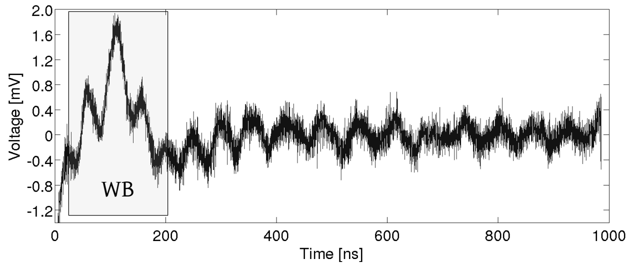


(a) Not encrypted

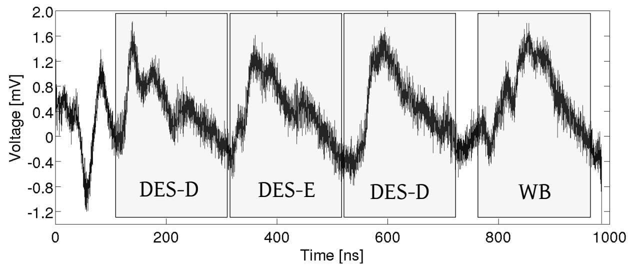


(b) Encrypted

Figure 3: Raw measurements of the power consumption at VCC_{INT} during the time period of the decryption



(a) Not encrypted



(b) Encrypted

Figure 4: Filtered power consumption measured at VCC_{INT} during the time period of the decryption.

of the measurements that has been discarded by the filter, i.e., containing the frequency components that have been blocked by the band block filter. We recall that, since differential power analysis relies on exploiting the differences in the power consumption caused by different inputs, the timewise variance computed over a significant number of traces is a reasonable index of the information contained in the signal. Figure 5 presents the computed variances for the time window pertaining the three decryption/encryption operations, based on processing 50, 000 different decryptions with different inputs. As we can see, the variance of the part of the signal kept by the filter features three distinct peaks at the beginning of each DES execution and is higher during the encryption process, with respect to the lower values assumed before and after. In contrast, the variance of the discarded signal is practically flat, suggesting that no information relevant to the decryption process has been omitted. Therefore it is conceivable that the oscillating power consumption is actually the power profile of the PowerPC 401 core embedded in the FPGA. The core has a reference working frequency of 300 MHz which seems compatible with our measurements.

The analysis suggest the presence of a large buffer employed to store the result of a whole DES encryption/decryption operation, due to the high variance at the beginning of every DES computation. It is also possible to infer the duration of a single DES round from the variance figure by simply computing the distance between the peaks: exactly one DES execution is encompassed between them. Through calculating this distance we obtain a computation time for the DES of 217 ns, corresponding to 651 ns for the execution of three full DES computations. This figure is well within the maximum bounds enforced by the JTAG input rate calculated in Sect. 3.1 which mandates the triple DES to be faster than 1.94 μs. The peak to peak amplitude of the decryption signal is roughly 2.8 mV, thus the measurement setup should have enough sensitivity to capture this signal.

3.3.1 Power-Profiling the Decryption Engine

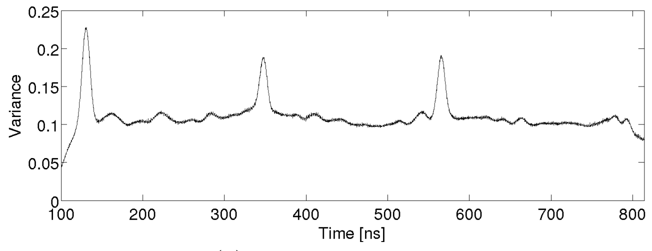
Performing a power analysis attack requires to know the architecture of the target device to select a power model which matches to its power consumption characteristics. This information is missing when attacking a black box. Having identified the point in time where the triple DES decryption is performed by the device we thus proceed with investigating the correct model for the power consumption. During this profiling phase, that aims at determining the underlying hardware structure of the decryption engine, the secret keys used for the 3DES are known. Accordingly, we are able to fully compute the intermediate results of the cipher in every encryption/decryption round.

We selected three different keys in the Xilinx ISE tools and generated the corresponding key file and an encrypted bitstream. After configuring the keys into the FPGA using a Xilinx programming device we collected 50, 000 power traces for the same number of ciphertexts (64-bit blocks)[1] while sending the encrypted bitstream to the target device by means of our customized configuration module. In the following, in order to perform a power analysis of the acquired measurements one needs to align them. For this purpose we applied the cross correlation alignment explained in Sect. 2.3.1 on the whole trace, after the filtering.

Section 3.3 pinpoints the relevant part of the configuration process, i.e., when the second bit of a 64-bit ciphertext packet is sent to the FPGA. The acquired power traces must hence be related to one of the previous ciphertexts. In order to determine which ciphertext belongs to which power trace we calculated the intermediate values of every DES round, considering the first ciphertext block in the bitstream and the known key as inputs.

The power consumption model for the analysis was chosen based on an assumption about the internal DES hardware, i.e, the common design choice that the state of the cipher is saved in a buffer in every round. Consequently, we computed the correlation coefficient between the processed measurements and the HD of two consecutive round outputs of the first DES decryption, which is a typical hypothetical power model when attacking hardware. We supposed different relations between the ciphertexts and power traces. For the hypothesis that the previously sent ciphertext is processed when sending the current 64-bit packet a strong correlation appears right after the trigger point (see Fig. 6). As a result, the missing link, i.e., which ciphertext corresponds to which decryption during the configuration, is found. To

[1]Configuring this FPGA, a maximum of around 70, 000 ciphertexts are available.


(a) Kept by the filter

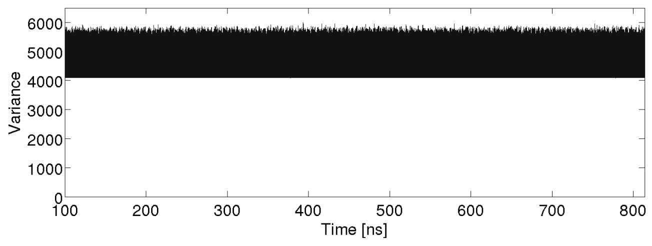

(b) Discarded by the filter

Figure 5: Variance of the power consumption for the part of the signal kept and discarded by the filter

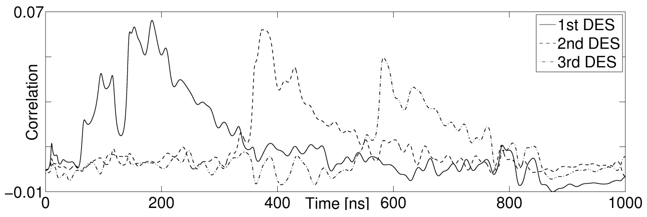


Figure 6: Correlation for hypothesizing the HD between the output of the first two rounds of each DES module

gain further insights into the hardware architecture we repeated the same for the input of the second and the third DES executions. The results are shown in Fig. 6: clearly, the correlation corresponding to each of the three DES runs can be spotted. In order to further confirm the presence of a buffer between every round of the DES cipher, we ran a series of power analyses, this time hypothesizing the HD between the input and the output of every DES round. Fig.7 depicts the correlation related to the 16 rounds at hand of the second DES execution. The correlation peak for the inner buffers is clearly present and the position of the peak shifts forward in time, increasing with the number of the hypothesized round. An important point here is the role of the initial permutation (IP) of DES. By trial-and-error we found that the reliable correlation results are obtained when excluding IP and IP^{-1} in the hypothesis. A self-evident hypothetical power model for attacking the first round of the first DES is $HD = HW(IP(C) \oplus Round(IP(C), K))$, where C is a 64-bit ciphertext, *Round* represents the 64-bit DES round function, and K is the last 48-bit subkey of the first DES. Accordingly, all the correlation results shown above base on this hypothetical power model.

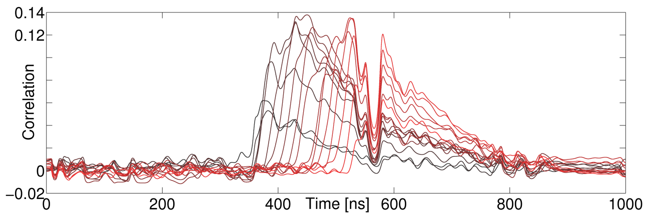


Figure 7: Correlation peaks for HD between the outputs of all the 16 rounds of the second DES

Observing that the features of the correlation figures have a large period in time, i.e., the peaks rise and fall relatively slowly, we tried a more restrictive filtering strategy: an additional low-pass filter with a cut-off frequency of 50 MHz was applied to all the collected traces and the HD-based correlation analysis was repeated. The results did not show any significant change in the shape and the amplitude of the correlations, thus we proceed in the actual attack with low-pass filtered traces.

3.4 The V2 DES Engine Internals

We conclude the profiling with some assumptions about the hardware implementation of the 3DES engine inside the Virtex-II Pro. It is conceivable that it employs a round function executing a round of the cipher per clock cycle since DES employs 8 different S-boxes that cannot be shared during a round computation. The correlation peaks in Fig. 7 moving forward in time with the targeted round of DES supports this assumption. The corresponding hypothetical architecture is depicted in Fig. 3.4. Due to the Feistel structure of the cipher, the encryption and decryption can be realized by 48 successive rounds, supplying the subkeys in the correct order, and applying the IP before the start of the first round and IP^{-1} after the last round.

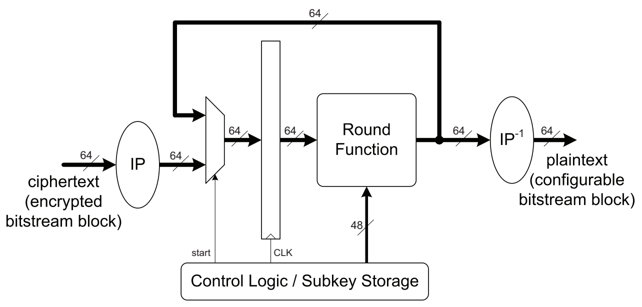

64
64
ciphertext (encrypted bitstream block)
IP
64
64
64
Round Function
64
IP^{-1}
64
plaintext (configurable bitstream block)
48
start
CLK
Control Logic / Subkey Storage

Figure 8: Our hypothetical architecture of the internal triple DES module of Virtex-II Pro FPGA

3.5 Extracting Keys from the Virtex 2

After choosing the proper information leakage model in the previous section, we now proceed to perform the attack in a real-world scenario: this time, an FPGA with an unknown set of keys is attacked. During a normal power-up of the target device, the adversary measures the power traces and collects the corresponding ciphertexts contained in the bitstream. Since the bitstream is encrypted with a secure cipher, even the simplest design results in the ciphertexts being uniformly distributed — a perfect condition for DPA. Similar to most of the side-channel attacks the attacker can now apply a divide-and-conquer approach by hypothesizing and correlating on small parts of the key, one part after the other, to finally obtain the full secret key. In the case of DES, each subkey of length 48-bit can be divided into 6-bit parts, since the value of a single bit of the output of a DES round is influenced by 6 key bits added to the input of the S-box computing it. To reveal the first round key the attack therefore has to be repeated eight times, once for each key portion considered in the attack. After the first 48 bits of the round key have been recovered, the remaining 6 bits of the 56-bit DES key are exposed by attacking the second round of the DES operation. We performed a full key recovery involving 50, 000 traces by hypothesizing the HD between a single bit of the output from a DES round and the previous value being stored in the round buffer (for simplicity consider a bit of the round buffer in Fig. 3.4). Once the 56-bit key for the first DES decryption is known the input for the subsequent DES encryption is computed and its 56-bit key (and finally that of the last DES decryption) is revealed similarly. To quantify the minimum sampling frequency required for a successful key recovery, we further investigated the fact

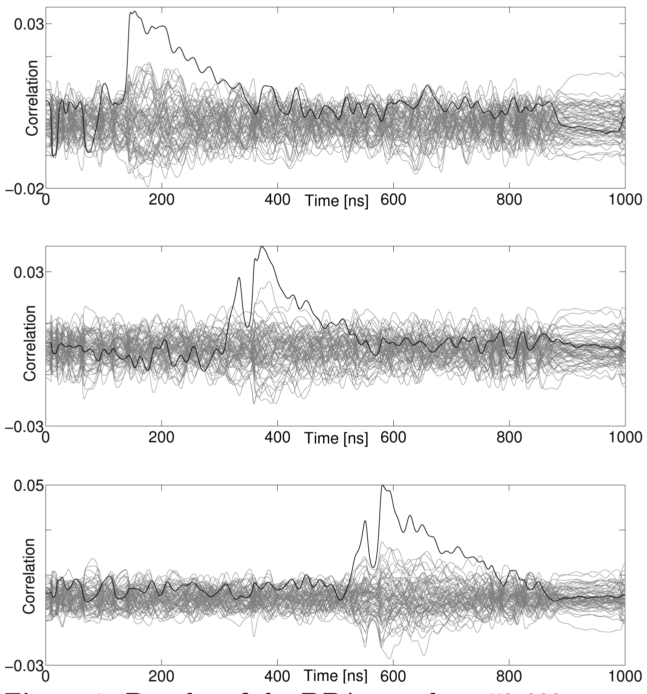


Figure 9: Results of the DPA attack on 50,000 traces employing non decimated lowpass traces targeting the (a) first (b) second (c) third DES execution (correct key guess in black, the others in gray)

that the attacks still works if the traces are sampled at a sensibly lower frequency. Following the fact that we applied a lowpass filter at 50 MHz, we applied a 100 times decimation to the traces, i.e., use only every hundredth sampling point and discard the rest, and repeated all the attacks in order to see if the reduced traces are still containing enough information. As a result, all the attacks mentioned up to now are repeatable on the decimated traces, thus yielding a significant speedup in the required computation time and a reduction in the memory fingerprint.

It is possible to recover a 6-bit DES subkey in less than 4 seconds of computation time on a common desktop PC, with a memory fingerprint smaller than 20 MB. The whole 112-bit key of a 2-key triple DES is hence revealed by our attack in roughly two minutes and the 168-bit key of a 3-key triple DES in three minutes. The trace decimation also impacts on the minimum sampling frequency required for the digital oscilloscope: since the decimated traces are now sampled at a rate of 100 MS/s this can be regarded as a new lower bound for the required sampling frequency of the oscilloscope.

3.5.1 *Practical Results*

So far, the same set of 50,000 power traces (acquired from loading a single bitstream into the FPGA) was used to profile the power consumption characteristics of the device and to perform the actual attacks. To determine the minimum number of traces required to reveal the complete triple DES key we have examined all subkeys of all three DES executions accordingly. As illustrated in Fig. 9 the power consumption right after the trigger signal is disturbed by the I/O activity, causing that a key-recovery attack on the first DES execution is harder than the others. The result of the attack over the number of traces for the worst case is shown by Fig. 10 indicating that with the current attack setup a minimum of 25,000 measurements is required.

The minimum number of required traces is largely affected by the measurement setup and environmental noise. Depending on these parameters therefore the attacker may need more traces. Our target FPGA, i.e., XC2VP7, is one of the smallest variants on the market. Still, the size of its configuration bitstream (independent of the size of the design) suffices to acquire more than 70,000 measurements of different ciphertexts being encrypted by the 3DES engine. Hence, loading of a single bitstream into our target FPGA during a power-up suffices to acquire close to three times more measurements than required for a successful recovery of the full secret key — this indicates that our attack can be conducted on real-world products even in a very noisy measurement environment. Note, that repeating the measurements for more power-ups does not increase the number of different ciphertexts — still averaging over the power-ups allows to reduce the noise level.

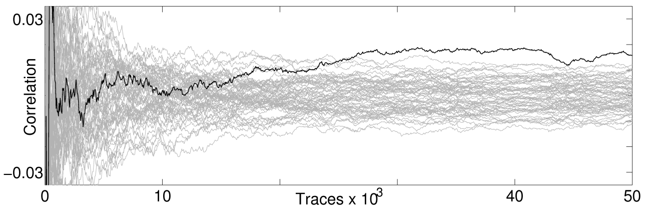


Figure 10: Result of the DPA attacks targeting the first DES as a function of number of traces (correct key guess in black, the others in gray)

4. IMPLICATIONS

In the previous section we have shown that the bitstream encryption feature aiming to provide IP protection can be circumvented by extracting the secret keys used for the encryption via power analysis. The complete content of any Virtex-II Pro protected with bitstream encryption can fall into the hands of a competitor or criminal — this may imply system-wide damage, if IP such as encryption schemes and keys programmed into the FPGA, or similar secrets, are misused or become public. The attacks hence have a devastating impact on the security of products employing bitstream encryption in the real-world.

For performing our key-recovery, an attacker needs knowledge about side-channel cryptanalysis, a basic lab measurement setup and physical access to the target device in which the Virtex-II Pro FPGA is embedded. The only modification required to be performed on the product to be attacked is removing the capacitances which have negative effect on power measurements. Also, the lithium battery must remain connected to the FPGA to ensure that the secret keys are not zeroed. The adversary connects her measurement equipment and powers up the product-under-attack once, while monitoring the loading of the encrypted bitstream. Finally, she performs the statistical evaluation on a standard PC to recover the full secret key of the 3DES. Applying the methods described in this paper, the keys used for the bitstream encryption can be recovered with modest efforts in less than one hour, using the measurements of only one single configuration process.

After the key recovery the attacker can decrypt the protected bitstream from the Virtex-II Pro FPGA, and hence possesses its complete content. As a consequence, she can produce a clone of the original FPGA on the basis of an empty device by simply configuring it with the extracted bitstream, even without using the encryption feature, for example to replicate the product of a competitor.

Furthermore, by reverse-engineering the internal wiring of the FPGA from the bitstream a stolen design can be analyzed, the internal secrets extracted and used for malicious purposes. The stolen design could even be improved and disguised as an own product that outrivals the original. While Xilnix tries to establish security by obscurity by keeping the exact mapping of the bitstream to the internal circuitry of their product confidential, there already exist methods to recover the original design of an FPGA from a bitstream [24, 30]. Amongst others, a free software project[2] currently supports reverse-engineering bitstreams of Virtex 2, Virtex 4 and Virtex 5 FPGAs.

In addition to lost IP, in a security-critical environment reprogramming the attacked FPGA with an ill-intended modified code, e.g., to accomplish malfunctioning or to unnoticedly implement a hardware trojan [21], is a particularly damaging option. In the following we illustrate possible implications that our attacks may have at hand of two examples from the commercial sector[3], without naming manufacturers.

4.1 Real-World Example 1: Set Top Box

Set-top boxes are widely used in the field for receiving digital TV and radio programmes, pay-per-view special events, allow to watch recent movies on a subscription basis and enable video on demand, i.e., remotely renting a video including the ability to pause, rewind, and fast forward. A set-top box is an ideal candidate for using FPGAs with bitstream encryption: In addition to the computationally highly demanding demodulating and decompressing algorithms for descrambling the transmitted TV and radio programmes, the manufacturers often employ their proprietary encryption schemes, e.g., for the above mentioned applications. The secure bitstream encryption feature enables to regularly update and improve the firmware running on the set-top boxes in the households, for example by cable or a satellite link, and establish corresponding business models without disclosing the descrambling schemes or secret keys to potential attackers.

Our key-recovery attack puts set-top boxes relying on the investigated or similar FPGAs using bitstream encryption at a high risk: an attacker extracting the firmware of a set-top box knows amongst others the scrambling scheme and secret keys, can gain access to all digital content without paying and is thus able to circumvent the above mentioned business models. A criminal can make high financial gain (and cause high financial losses for the service provider) by producing new set-top boxes or modifying the firmware of existing set-top boxes to enable the usage of the services free of cost by the customers.

4.2 Real-World Example 2: Router

Network routers redirect data traffic with a speed of hundreds of Gigabits per second in local networks and often constitute the interface to the outside world, e.g., the Internet, for companies, government agencies and private households. In addition to routing data, an FPGA in common products often realizes security-relevant functions such as a firewall to separate private local networks from other insecure ones. In case of a bug in the firmware or if a security fix is required, some routers can be updated remotely — again, the bitstream encryption is the basis for establishing this functionality. Assuming a router that is based on a Virtex-II Pro, an attacker having one-time physical access to it can perform our key-recovery attack. From there on, she can remotely initiate firmware updates, e.g., via the Internet, to modify the functionality of the router according to her demands.

A denial-of-service attack, aiming at malfunctioning or to destroy the device or other connected equipment is one option. More likely, a modified firmware can be used to open a covered channel allowing the adversary to spy out and access data from the internally connected computers and other devices, e.g., by implementing a trojan horse that secretly forwards all internal traffic to the adversary. The user typically has no means to verify whether his firmware has been manipulated or if it is original, hence — depending on the location where the manipulated router is used — a modified firmware can have a devastating impact on the privacy and security, and even the safety, of the attacked individual or company.

4.3 Scope of our Findings

As researchers of a university with this paper we intend to inform about possible vulnerabilities when using bitstream encryption and warn the end-users of products incorporating this feature from possible damage. We believe there can be fairly severe real-world implications (depending on the commercial devices in which Virtex-II Pro FPGAs are used) due to our findings. Furthermore, it seems highly likely that certain determined attackers, e.g., foreign intelligent services, are already capable of extracting and altering FPGA designs using power analysis techniques. Since the success of a power analysis attack in general doesn't depend on the cipher employed, it is conceivable that similar attacks can be applied to newer generations of FPGAs from different manufacturers, employing different ciphers, such as the Virtex 4 family or Altera products using AES-256 in their bit encryption schemes [29, 8].

5. CONCLUSION

We presented the first attacks targeting the bitstream encryption of FPGAs in the literature. By profiling the power consumption of the target FPGA we reverse-engineered all relevant details of the security feature and pinpointed the time instant when the decryption of the ciphertext blocks of the bitstream is performed by a dedicated hardware contained in the FPGA. We identified the appropriate power model for attacking the triple DES engine by means of power analysis and deduced the internal architecture of the hardware. Our proposed techniques for the digital signal processing of the power measurements enable a highly efficient power analysis attack that allows the full secret key of the triple DES to be extracted in only 3 minutes of computation time from only $25,000$ measurements, obtainable with a single boot up of the device. Our findings indicate that the attacks are possible using a low-cost oscilloscope with a sample rate as little as 100 MS/s.

In particular, we have successfully exemplified our attack at hand of the bitstream decryption of a Virtex-II Pro XC2CP7 FPGA manufactured by the market leader Xilinx. We are able to recover all three different keys used by its triple DES module from a single power-up of the device in a real-world scenario. The lithium battery for the key storage, providing extra security according to Xilinx, does not have to be removed for the attack. It is highly conceivable that similar attacks can be applied to other series of FPGAs, e.g., from other manufacturers.

Besides the obscurity that emerges from Xilinx keeping the details of their bitstream files secret we encountered no countermeasures against side-channel analysis — an alarming fact considering the protection of many devices deployed in the field: an attacker knowing the secret key used for the encryption of the bitstream can obtain all secrets and intellectual property contained in commercial products, e.g., proprietary encryption schemes or processing algorithms.

As a consequence of our attacks, cloning of the products protected by the bitstream encryption scheme becomes straightforward. After reverse engineering the content of the FPGA from the bitstream, improved products could be marketed by a competitor that outrival the original. Much worse, the content of commercial products can be updated remotely, e.g., via Internet, with a maliciously modified new firmware. This poses a severe threat to the reliability of the products, puts the privacy of individuals and companies at a high risk and further enables to infect the devices with embedded malware.

[2]www.ulogic.org

[3]for obvious reasons we do not cover military applications here.

6. REFERENCES

[1] Defense Science Board. http://www.acq.osd.mil/dsb/.

[2] Maxtor Basics Personal Storage 3200. http://www.seagate.com/www/en-us/support/downloads/personal_storage/ps3200-sw.

[3] Side-channel Attack Standard Evaluation Board (SASEBO). Further information are available via http://www.rcis.aist.go.jp/special/SASEBO/.

[4] Xilinx ISE Design Suite. http://www.xilinx.com/tools/designtools.htm.

[5] IEEE Standard Test Access Port and Boundary-Scan Architecture. *IEEE Std 1149.1-2001*, pages i –200, 2001.

[6] D. Abraham, G. Dolan, G. Double, and J. Stevens. Transaction Security System. In *IBM Systems Journal 30*, pages 206–229, 1991.

[7] G. Agosta, A. Barenghi, F. D. Santis, and G. Pelosi. Record Setting Software Implementation of DES Using CUDA. *Information Technology: New Generations, Third International Conference on*, pages 748–755, 2010.

[8] ALTERA. Using the Design Security Feature in Stratix II and Stratix II GX Devices (AN 341 version 2.3). Technical report, August 2009. http://www.altera.com/literature/an/an341.pdf.

[9] ATK. XM25 Counter Defilade Target Engagement System. http://www.atk.com/customer_solutions_missionsystems/documents/sw_iw_xm25.pdf, May 2009. Post "FPGAs in interesting places – the XM25 Airburst Weapon System" by Saar Drimer on www.fpgasecurity.com.

[10] A. Barenghi, G. Pelosi, and Y. Teglia. Improving first order differential power attacks through digital signal processing. In *ACM-SIGSAC International Conference on Security of Information and Networks*, pages 124–133. ACM, 2010.

[11] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *CHES*, pages 16–29, 2004.

[12] W. J. Broad, J. Markoff, and D. E. Sanger. Israeli Test on Worm Called Crucial in Iran Nuclear Delay. Technical report, New York Times, January 2011. http://www.nytimes.com/2011/01/16/world/middleeast/16stuxnet.html.

[13] O. Coudert. Why FPGA startups keep failing, 2009. FPGA market shares according to Gartner Inc, 2008.

[14] S. Drimer. Security for volatile FPGAs. Technical Report UCAM-CL-TR-763, University of Cambridge, Computer Laboratory, Novembre 2009. ISSN 1476-2986 http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-763.pdf.

[15] 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 2008*, volume 5157 of *LNCS*, pages 203–220. Springer.

[16] Eric Peeters and François-Xavier Standaert and Jean-Jacques Quisquater. Power and Electromagnetic Analysis: Improved Model, Consequences and Comparisons. *Integr. VLSI J.*, 40(1):52–60, 2007.

[17] T. Güneysu, T. Kasper, M. Novotný, C. Paar, and A. Rupp. Cryptanalysis with COPACOBANA. *IEEE Transactions on Computers*, 57(11):1498–1513, 2008.

[18] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *CRYPTO 99*, pages 388–397. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1666.

[19] R. Krueger. Application Note XAPP766: Using High Security Features in Virtex-II Series FPGAs. Technical report, XILINX, 2004. http://www.xilinx.com/support/documentation/application_notes/xapp766.pdf.

[20] A. Lesea. IP Security in FPGAs, White Paper WP 261. Technical report, XILINX, February 2007.

[21] L. Lin, M. Kasper, T. Güneysu, C. Paar, and W. Burleson. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *CHES*, volume 5747 of *LNCS*, pages 382–395. Springer, 2009.

[22] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[23] NIST. FIPS-46-3: Data Encryption Standard (DES), 1999.

[24] J.-B. Note and É. Rannaud. From the bitstream to the netlist. In M. Hutton and P. Chow, editors, *16th International Symposium on Field Programmable Gate Arrays, FPGA 2008*. ACM, 2008.

[25] S. B. Örs, E. Oswald, and B. Preneel. Power-Analysis Attacks on an FPGA - First Experimental Results. In *CHES 2003*, volume 2779 of *LNCS*, pages 35–50. Springer, 2003.

[26] Recurity Labs. Embedded Analysis. 27th Chaos Communication Congress, Dec. 2010. http://events.ccc.de/congress/2010/wiki/Embedded_Analysis.

[27] F.-X. Standaert, S. B. Örs, J.-J. Quisquater, and B. Preneel. Power Analysis Attacks Against FPGA Implementations of the DES. In *Field Programmable Logic and Application - FPL 2004*, volume 3203 of *LNCS*, pages 84–94. Springer, 2004.

[28] A. Telikepalli. Is Your FPGA Design Secure? XCell Journal, XILINX, Fall 2003.

[29] C. W. Tseng. Lock Your Designs with the Virtex-4 Security Solution. XCell Journal, XILINX, Spring 2005.

[30] T. J. Wollinger, J. Guajardo, and C. Paar. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Transactions in Embedded Computing Systems (TECS)*, 3(3):534–574, 2004.

[31] XILINX. Virtex-2 Platform FPGA User Guide (UG002 version 2.2). Technical report, November 2007. http://www.xilinx.com/support/documentation/user_guides/ug002.pdf.

[32] XILINX. Virtex-II Pro and Virtex-II Pro X FPGA User Guide. Technical report, 2007. http://www.xilinx.com/support/documentation/user_guides/ug012.pdf.

[33] XILINX. Virtex-II Pro Platform FPGAs: Complete Data Sheet (DS 083 version 4.7). Technical report, November 2007. http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf.