How Secure Are FPGAs in Cryptographic Applications? (Long Version) *

Thomas Wollinger and Christof Paar

Chair for Communication Security (COSY)
Horst Görtz Institute for IT Security
Ruhr-Universität Bochum, Germany
{wollinger, cpaar}@crypto.rub.de

Abstract. The use of FPGAs for cryptographic applications is highly attractive for a variety of reasons but at the same time there are many open issues related to the general security of FPGAs. This contribution attempts to provide a state-of-the-art description of this topic. First, the advantages of reconfigurable hardware for cryptographic applications are discussed from a systems perspective. Second, potential security problems of FPGAs are described in detail, followed by a proposal of a some countermeasure. Third, a list of open research problems is provided. Even though there have been many contributions dealing with the algorithmic aspects of cryptographic schemes implemented on FPGAs, this contribution appears to be the first comprehensive treatment of system and security aspects.

Keywords: cryptography, FPGA, security, attacks, reconfigurable hardware, physical

1 Introduction

The choice of the implementation platform of a digital system is driven by many criteria and heavily dependent on the application area. In cryptographic applications there are many criteria which are unique to the security context in which there are used. In addition to the aspects of algorithm and system speed and costs — which are present in most other application domains too — there are crypto-specific ones: physical security (e.g., against key recovery and algorithm manipulation), flexibility (regarding algorithm parameter, keys, and the algorithm itself), power consumption (absolute usage and prevention of power analysis attacks), and other side channel leakages.

The advantages of software implementations include ease of use, ease of upgrade, portability, low development costs, low unit prices and flexibility. On the down side, a software implementation offers moderate speed, high power consumption compared to custom hardware, and only limited physical security,

* This research was partially sponsored by the German Federal Office for Information Security (BSI).

especially with respect to key storage [Sch96]. ASIC implementations show lower price per unit, can reach high speeds, and can have low power dissipation. Furthermore hardware implementations of cryptographic algorithms are more secure because they cannot as easily be read or modified by an outside attacker [Dou99] as software implementations. The downside of ASIC implementations, however, are higher development costs and the lack of flexibility with respect to algorithm and parameter switching. Reconfigurable hardware devices, such as Field Programmable Gate Arrays (FPGAs), seem to combine the advantages of SW and HW implementations. At the same time, there are still many open questions regarding FPGAs as a module for security functions. There has been a fair amount of work been done by the research community dealing with the algorithmic and computer architecture aspects of crypto schemes implemented on FPGAs since the mid-1990s (see, e.g., relevant articles in [KP99,KP00,KNP01,KKP02]), often focusing on high-performance implementations. At the same time, however, very little work has been done dealing with the system and physical aspects of FPGAs as they pertain to cryptographic applications. It should be noted that the main threat to a cryptographic scheme in the real world is *not* the cryptanalysis of the actual algorithm (such as the Enigma break by the allied intelligence services during WW II), but rather the exploration of weaknesses of the implementation. Given this fact, we hope that the contribution at hand is of interest to readers in academia, industry and government sectors.

In this paper we'll start in Section 2 with the description of the advantages of FPGAs in cryptographic applications from a systems perspective. Then, we highlight important questions pertaining to the *security* of FPGAs when used for crypto algorithms in Section 3. A major part of this contribution is a state-of-the-art perspective of security issues with respect to FPGAs, by illuminating this problem from different viewpoints and by trying to transfer problems and solutions from other hardware platforms to FPGAs (Section 4). In Section 5, we provide a list of open problems. Finally, we end this contribution with some conclusions. We would like to stress that this contribution is not based on any practical experiments, but on a careful analysis of available publications in the literature and on our experience with implementing crypto algorithms.

2 System Advantages of FPGAs for Cryptographic Applications

In this section we list the potential advantages of reconfigurable hardware (RCHW) in cryptographic applications. Some of the ideas are taken from our earlier work in [EYCP01] which have been extended.

Algorithm Agility: This term refers to the switching of cryptographic algorithms during operation of the targeted application. One can observe that the majority of modern security protocols, such as IPSec [KA98], SSL [FKK96] or TLS [DA99], are algorithm independent and allow for multiple encryption algorithms. The encryption algorithm is negotiated on a per-session basis and a wide variety may be required; e.g., IPsec allows among others DES, 3DES, Blowfish, CAST, IDEA, RC4 and RC6 as algorithms, and future extensions are possible. Whereas algorithm agility is costly with traditional hardware, FPGAs can be reprogrammed on-the-fly.

Algorithm Upload: It is perceivable that fielded devices are upgraded with a new encryption algorithm. Algorithm upload can be necessary because a current algorithm was broken (e.g., Data Encryption Standard - DES [Fed77]), a standard expired (e.g. DES), a new standard was created (e.g. Advanced Encryption Standard - AES [U.S01]), and/or that the list of ciphers in an algorithm independent protocol was extended. Assuming there is some kind of (temporary) connection to a network such as the Internet, FPGA-equipped encryption devices can upload the new configuration code. Notice that the upgrade of ASIC-implemented algorithms is practically infeasible if many devices are affected or if the systems are not easily accessible, for instance in satellites.

Architecture Efficiency: In certain cases a hardware architecture can be much more efficient if it is designed for a specific set of parameters. Parameters for cryptographic algorithms can be for example the key, the underlying finite field, the coefficient used, and so on. Generally speaking, the more specific an algorithm is implemented the more efficient it can become. An efficient parameter-specific implementation of the symmetric cipher IDEA [LM90,LMM91] was presented in [TG99]. The general modular multiplication in IDEA requires 16 partial multiplications and only eight assuming a fixed key. Another example taken from asymmetric cryptography is the arithmetic architectures for Galois fields. Squaring in $GF(2^m)$ takes $m/2$ cycles with a general architecture, but only one cycle if the architecture is compiled for a fixed field [Wu99]. FPGAs allow this type of design and optimization with specific parameter set. Due to the nature of FPGAs, the application can be changed totally or partially.

Resource Efficiency: The majority of security protocols are hybrid protocols, e.g. IPSec, SSL, and TLS. This implies that a public-key algorithm is used to transmit the session key. After the key was established a private-key algorithm is needed for data encryption. Since the algorithms are not used simultaneously, the same FPGA device can be used for both through runtime reconfiguration.

Algorithm Modification: There are applications which require modification of standardized cryptographic algorithms, e.g., by using proprietary S-boxes or permutations. Such modifications are easily made with RCHW. One example, where a standardized algorithm was slightly changed, is the UNIX password encryption [MvOV97] where DES is used 25 times in a row and a 12-bit salt modifies the expansion mapping Furthermore, in many occasions cryptographic primitives or their modes of operation have to be modified according to the application.

Throughput: General-purpose CPUs are not optimized for fast execution especially in the case of public-key algorithms. Mainly because they lack instructions for modular arithmetic operations on long operands, for example exponentiation for RSA [RSA78]. Although typically slower than ASIC implementations, FPGA implementations have the potential of running substantially faster than software implementations. The block cipher AES, exemplarily, reaches a data rate of 112,3 Mbit/s and 718,4 Mbit/s on a DSP TI TMS320C6201 [WWGP00] and Pentium III [Lip], respectively. In comparison, the FPGA implementation of the same algorithm on a Virtex XCV-1000BG560-6 achieved 12 GBit/s using 12,600 slices and 80 RAMs [GC01]. On the other hand, an ASIC encrypts at about double the speed of the FPGA, e.g. Amphion CS5240TK reaches 25.6 Gbit/s at 200MHz [Amp].

Cost Efficiency: There are two cost factors that have to be taken into consideration, when analyzing the cost efficiency of FPGAs: cost of development and unit prices. The costs to develop an FPGA implementation of a given algorithm are much lower than for an ASIC implementation, because one is actually able to use the given structure of the FPGA (e.g. look-up table) and one can test the reconfigured chip endless times without any further costs. This results in a shorter time-to-market period, which is nowadays an important cost factor. The unit prices are not so significant when comparing them with the development costs. However, for high-volume applications, ASIC solutions usually become the more cost-efficient choice.

We would like to stress that the advantages above are not necessarily restricted to cryptographic applications. They can also be exploited in different contexts. For example, the remote upload of a configuration can be used to fix bugs in fielded devices, or to upgrade existing devices to make them compatible with new standards. However, some of the cryptographic advantages, such as replacement of a broken algorithm, may carry more weight here than in other application domain. Note that the listed potential advantages of FPGAs for cryptographic applications can only be exploited if the security shortcomings of FPGAs discussed in the following have been addressed.

3 Security Shortcomings of FPGAs

This section summarizes security problems produced by attacks against given FPGA implementations. First we would like to state what the possible goals of such attacks are.

3.1 Objectives of an Attacker

The most common threat against an implementation of cryptographic algorithm is to learn a confidential cryptographic key, that is, either a symmetric key or the private key of an asymmetric algorithm. Given that the algorithms applied are publicly known in most commercial applications, knowledge of the key enables the attacker to decrypt future (assuming the attack has not been detected and countermeasures have not been taken) and, often more harming, past communication which had been encrypted. Another threat is the one-to-one copy, or "cloning", of a cryptographic algorithm *together* with its key. In some cases it can be enough to run the cloned application in decryption mode to decipher past and future communication. In other cases, execution of a certain cryptographic operation with a presumingly secret key is in most applications the sole criteria which authenticates a communication party. An attacker who can perform the same function can masquerade as the attacked communication party. Yet another threat is given in applications where the cryptographic algorithms are proprietary. Even though such an approach is not wide-spread, it is standard practice in applications such as pay-TV and in government communication. In such scenarios it is already interesting for an attacker to reverse-engineer the encryption algorithm itself. The associated key might later be recovered by other methods (e.g., bribery or classical cryptanalysis.)

The discussion above assumes mostly that an attacker has physical access to the encryption device. Whether that is the case or not depends heavily on the application. However, we believe that in many scenarios such access can be assumed, either through outsiders or through dishonest insiders.

In the following we discuss vulnerabilities of modern FPGAs against such attacks. In areas were no attacks on FPGAs have been published, we tried to extrapolate from attacks on other hardware platforms, mainly memory cell and chip cards.

3.2 Black Box Attack

The classical method to reverse engineer a chip is the so called Black Box attack. The attacker inputs all possible combinations, while saving the corresponding outputs. The intruder is then able to extract the inner logic of the FPGA, with the help of the Karnaugh map or algorithms that simplify the resulting tables. This attack is only feasible if a small FPGA with explicit inputs and outputs is attacked and a lot of processor power is available. The reverse engineering effort grows and it will become less feasible as the size and complexity of the FPGA increases and with the usage of state machines, LFSRs (Linear Feedback Shift Registers), integrated storage, and so on [Dip].

3.3 Readback Attack

Readback is a feature that is provided for most FPGA families. This feature allows to read a configuration out of the FPGA for easy debugging. The idea of the attack is to read the configuration of the FPGA through the JTAG or programming interface in order to obtain secret information (e.g. keys) [Dip]. The readback functionality can be prevented with security bits provided by the manufactures. In [AEC99], the idea of using a security antifuse to prevent readout of information is patented.

However, it is conceivable, that an attacker can overcome these countermeasures in FPGA with fault injection. This kind of attack was first introduced in [BDL97] and it was shown how to break public-key algorithms by exploiting hardware faults. This publication, was followed by [BS97], where the authors introduced differential fault analysis, which can potentially be applied against all symmetric algorithms in the open literature. Meanwhile there have been many publications that show different techniques to insert faults, e.g., electro magnetic radiation [QS01], infrared laser [Ajl95], or even a flash light [SA02]. It seems very likely that these attacks can be easily applied to FPGAs, since they are not especially targeted to ASICs. If this is in fact feasible, an attacker is able to deactivate security bits and/or the countermeasures, resulting in the ability to read out the configuration of the FPGA [Kes,Dip].

3.4 Cloning of SRAM FPGAs

In a standard scenario, the configuration data is stored (unprotected) externally in nonvolatile memory (e.g., PROM) and is transmitted to the FPGA at power-up in order to configure the FPGA. An attacker could easily eavesdrop on the transmission and get the configuration file. This attack is therefore feasible for large organizations as well as for those with low budgets and modest sophistication.

3.5 Reverse-Engineering of the Bitstreams

The attacks described so far output the bitstream of the FPGA design. In order to get the design of proprietary algorithms or the secret keys, one has to reverse-engineer the bitstream. The condition to launch the attack is that the attacker has to be in possession of the (unencrypted) bitstream.

FPGA manufactures claim that the security of the bitstream relies on the disclosure of the layout of the configuration data. This information will only be made available if a non-disclosure agreement is signed, which is, from a cryptographic point of view, an extremely insecure situation. This security-by-obscurity approach was broken at least ten years ago when the CAD software company NEOCad reverse-engineered a Xilinx FPGA. NEOCad was able to reconstruct the necessary information about look-up tables, connections, and storage elements [Sea]. Hence, NEOCad was able to produce design software without signing non-disclosure agreements with the FPGA manufacturer. Even though a big effort has to be made to reverse engineer the bitstream, for large organizations it is quite feasible. In terms of government organizations as attackers, it is also possible that they will get the information of the design methodology directly from the vendors or companies that signed NDAs.

3.6 Physical Attack

The aim of a physical attack is to investigate the chip design in order to get information about proprietary algorithms or to determine the secret keys by probing points inside the chip. Hence, this attack targets parts of the FPGA, which are not available through the normal I/O pins. This can potentially be achieved through visual inspections and by using tools such as optical microscopes and mechanical probes. However, FPGAs are becoming so complex that only with advanced methods, such as Focused Ion Beam (FIB) systems, one can launch such an attack. To our knowledge, there are no countermeasures to protect FPGAs against this form of physical threat. In the following, we will try to analyze the effort needed to physically attack FPGAs manufactured with different underlying technologies.

SRAM FPGAs: Unfortunately, there are no publications available that accomplished a physical attack against SRAM FPGAs. This kind of attack is only treated very superficially in a few articles, e.g. [Ric98]. In the related area of SRAM memory, however there has been a lot of effort by academia and industry to exploit this kind of attack [Gut96,Gut01,AK97,WKM$^+$96,Sch98,SA93,KK99]. Due to the similarities in structure of the SRAM memory cell and the internal structure of the SRAM FPGA, it is most likely that the attacks can be employed in this setting.

Contrary to common wisdom, the SRAM memory cells do not entirely loose the contents when power is cut. The reason for these effects are rooted in the physical properties of semiconductors (see [Gut01] for more details). The physical changes are caused mainly by three effects: electromigration, hot carriers, and ionic contamination. Most publications agree that device can be altered, if 1) threshold voltage has changed by 100mV or 2) there is a 10% change in transconductance, voltage or current.

One can find attacks against SRAM memory cells using the access points provided by the manufactures. An extreme case of data recovery, was described in [AK97]. The authors were able to extract a DES master key from a module used by a bank, without any special techniques or equipment on power-up. The reason being that the key was stored in same SRAM cells over a long period of time. "I_{DDQ} testing" is one of the widely used methods to analyze SRAM cells and it is based on the analysis of the current usage. The idea is to execute a set of test vectors until a given location is reached, at which point the device current is measured. Hot carrier effects, cell charge, and transitions between different states can then be detected at the abnormal I_{DDQ} characteristic [Gut01,WKM$^+$96,Sch98]. Another possibilities for the attack are also to use the scan path that the IC manufacturers insert for test purposes or techniques like bond pad probing [Gut01].

When it becomes necessary to use access points that are not provided by the manufacturer, the layers of the chip have to be removed. Mechanical probing with tungsten wire with a radius of $0,1-0,2\mu m$ is the traditional way to discover the needed information. These probes provide gigahertz bandwidth with $100fF$ capacitance and $1M\Omega$ resistance. Due to the complex structure and the multi layer production of chips the mechanical testing is not sufficient enough. Focused Ion Beam (FIB) workstations can expose buried conductors and deposit new probe points. The functionality is similar to an electron microscope and one can inspect structures down to 5nm [KK99]. Electron-beam tester (EBT) is another measurement method. An EBT is a special electron microscope that is able to speed primary electrons up to 2.5 kV at 5nA. EBT measures the energy and amount of secondary electrons that are reflected.

Resulting from the above discussion of attacks against SRAM memory cells, it seems likely that a physical attack against SRAM FPGAs can be launched successfully, assuming that the described techniques can be transfered. However, the physical attacks are quite costly and having the structure and the size of state-of-the-art FPGA in mind, the attack will probably only be possible for large organizations, for example intelligence services.

Antifuse FPGAs: To discuss physical attacks against antifuse (AF) FPGAs, one has to first understand the programming process and the structure of the cells. The basic structure of an AF node is a thin insulating layer (smaller than $1 \mu m^2$) between conductors that are programmed by applying a voltage. After applying the voltage, the insulator becomes a low-resistance conductor and there exists a connection (diameter about 100nm) between the conductors. The programming function is permanent and the low-impedance state will persist indefinitely.

In order to be able to detect the existence or non-existence of the connection one has to remove layer after layer, or/and use cross-sectioning. Unfortunately, no details have been published regarding this type of attack. In [Dip], the author states that a lot of trial-and-error is necessary to find the configuration of one cell and that it is likely that the rest of the chip will be destroyed, while analyzing one cell. The main problem with this analysis is that the isolation layer is much smaller than the whole AF cell. One study estimates that about 800,000 chips with the same configuration are necessary to explore one configuration file of an Actel A54SX16 chip with 24,000 system gates [Dip]. Another aggravation of the attack is that only about 2–5 % of all possible connections in an average design are actually used. In [Ric98] a practical attack against AF FPGAs was performed and it was possible to alter one cell in two months at a cost of $1000. Based on these arguments some experts argue that physical attacks against AF FPGAs are harder to perform than against ASICs [Act02]. On the other hand, we know that AF FPGAs can be easily attacked without being connected to power. Hence, it is easier to drill holes to disconnect two connections or to repair destroyed layers.

Flash FPGAs: The connections in flash FPGAs are realized through flash transistors. That means the amount of electrons flowing through the gate changes after configuration and there are no optical differences as in the case of AF FPGAs. Flash FPGAs can be analyzed by placing the chip in a vacuum chamber and powering it up. The attacker can then use a secondary electron microscope to detect and display emissions. The attacker has to get access to the silicon die, by removing the packet, before he can start the attack [Dip]. However, experts are not certain about the complexity of such an attack and there are controversial discussions about its practicality [Act02,Ric98]

Other possible attacks against flash FPGAs can be found in the related area of flash memory. The number of write/erase cycles are limited to 10,000 – 100,000, because of the accumulation of electrons in the floating gate causing a gradual rise of the transistors threshold voltage. This fact increases the programming time and eventually disables the erasing of the cell [Gut01]. Another less common failure is the programming disturbance in which unselected erased cells gain charge when adjacent selected cells are written [ASH+93,Gut01]. Furthermore, electron emission causes a net charge loss. The electrons in the floating gate migrate to the interface with the underlying oxide from where they tunnel into the substrate [PGP+91]. In addition, hot carrier effects have a high influence, by building a tunnel between the bands. This causes a change in the threshold voltage of erased cells and it is especially significant for virgin cells [HCSL89]. Another phenomenon is overerasing, where an erase cycle is applied to an already-erased cell leaving the floating gate positively charged [Gut01].

All the described effects change in a more or less extensive way the cell threshold voltage, gate voltage, or the characteristic of the cell. We remark that the stated phenomenons apply as well for EEPROM memory and that due to the structure of the FPGA cell these attacks can be simply adapted to attack flash/EEPROM FPGAs.

3.7 Side Channel Attacks

Any physical implementation of a cryptographic system might provide a *side channel* that leaks unwanted information. Examples for side channels include in particular: power consumption, timing behavior, and electromagnet radiation. Obviously, FPGA implementations are also vulnerable to these attacks. In [KJJ99] two practical attacks, Simple Power Analysis (SPA) and Differential Power Analysis (DPA) were introduced. The power consumption of the device while performing a cryptographic operation was analyzed in order to find the secret keys from a tamper resistant device. The main idea of DPA is to detect regions in the power consumption of a device which are correlated with the secret key. Moreover, in some cases little or no information about the target implementation is required. Since their introduction, there has been a lot of work improving the original power attacks (see, e.g., relevant articles in [KP99,KP00,KNP01,KKP02]). More recently the first successful attacks based on the analysis of electromagnetic emissions have also been published [AARR]. Even though most of the published attacks are not specific to a particular platform, there has usually been an assumption that the underlying platform is either software or an ASIC. There seems to be very little work at the time of writing addressing the feasibility of actual side channel attacks against FPGAs. However, it seems almost certain that the different side channels can be exploited in the case of FPGAs as well.

4 How to Prevent Possible Attacks?

This section shortly summarizes possible countermeasures that can be provided to minimize the effects of the attacks mentioned in the previous section. Most of them have to be realized by design changes through the FPGA manufacturers, but some could be applied during the programming phase of the FPGA.

Preventing the Black Box Attack: The Black Box Attack is not a real threat nowadays, due to the complexity of the designs and the size of state-of-the-art FPGAs (see Section 3.2). Furthermore, the nature of cryptographic algorithms prevents the attack as well. Cryptographic algorithms can be segmented in two groups: symmetric-key and public-key algorithms. Symmetric key algorithms can be further divided into stream and block ciphers. Todays stream ciphers output a bit stream, with a period length of 128 bits [TABG03]. Block ciphers, like AES, are designed with a block length of 128 bits and a minimum key length of 128 bits. Minimum length in the case of public-key algorithms is 160 bits for elliptic curve cryptosystems [Mil86,Kob87]) and 1024 bits for discrete logarithm and RSA-based systems. It is widely believed that it is infeasible to perform a brute force attack and search a space with 2^{80} possibilities. Hence, implementations of this algorithms can not be attacked with the black box approach.

Preventing the Cloning of SRAM FPGAs: There are many suggestions to prevent the cloning of SRAM FPGAs, mainly motivated by the desire to prevent reverse engineering of general, i.e., non-cryptographic, FPGA designs. One solution would be to check the serial number before executing the design and delete the circuit if it is not correct. This approach is not practical because of the following reasons: 1) The whole chip, including the serial number can be easily copied; 2) Every board would need a different configuration; 3) Logistic complexity to manage the serial numbers [Kes]. Another solution would be to use dongles to protect the design (a survey on dongles can be found in [Kea01]). Dongles are based on security-by-obscurity, and therefore do not provide solid security. A more realistic solution would be to have the nonvolatile memory and the FPGA in one chip or to combine both parts by covering them with epoxy. This reflects also the trend in chip manufacturing to have different components combined, e.g., the FPSLIC from Atmel. However, it has to be guaranteed that an attacker is not able to separate the parts.

Encryption of the configuration file is the most effective and practical countermeasure against the cloning of SRAM FPGAs. There are several patents that propose different scenarios related to the encryption of the configuration file: how to encrypt, how to load the file into the FPGA, how to provide key management, how to configure the encryption algorithms, and how to store the secret data [Jef02,Aus95,Eri99,SW99,Alg]. Furthermore, there are a good number of publications that suggest encryption schemes, e.g. [YN00] proposes a partial encryption of the bitstream and [KB00] published the idea to divide the configuration file and encrypt the parts with different keys. The 60RS family from Actel was the first attempt to have a key stored in the FPGA in order to be able to encrypt the configuration file before transmitting it to the chip. The problem was that every FPGA had the same key on board. This implies that if an attacker has one key he can get the secret information from all FPGAs.

An approach in a completely different direction would be to power the whole SRAM FPGA with a battery, which would make transmission of the configuration file after a power loss unnecessary. This solution does not appear practical, however, because of the power consumption of FPGAs. Hence, a combination of encryption and battery power provides a possible solution. Xilinx addresses this with an on-chip 3DES decryption engine in its Virtex II [Xil] (see also [PWF+00]), where the two keys are stored in the battery powered memory.

Preventing the Physical Attack: To prevent physical attacks, one has to make sure that the retention effects of the cells are as small as possible, so that an attacker can not detect the status of the cells. Already after storing a value in a SRAM memory cell for 100–500 seconds, the access time and operation voltage will change [vdPK90]. Furthermore, the recovery process is heavily dependant on the temperature: 1.5 hours at $75°C$, 3 days at $50°C$, 2 month at $20°C$, and 3 years at $0°C$ [Gut01]. The solution would be to invert the data stored periodically or to move the data around in memory. Cryptographic applications cause also long-term retention effects in SRAM memory cells by repeatedly feeding data through the same circuit. One example is specialized hardware that always uses the same circuits to feed the secret key to the arithmetic unit [Gut01]. Neutralization of this effect can be achieved by applying an opposite current [TCH93] or by inserting dummy cycles into the circuit [Gut01]. In terms of FPGA application, it is very costly or even impractical to provide solutions like inverting the bits or changing the location for the whole configuration file. A possibility could be that this is done only for the crucial part of the design, like the secret keys. Counter techniques such as dummy cycles and opposite current approach can be carried forward to FPGA applications.

Antifuse FPGAs can only be protected against physical attack, by building a secure environment around them. If an attack was detected every cell should be programmed in order not to leak any information or the antifuse FPGA has to be destroyed.

In terms of flash/EEPROM memory cell, one has to consider that the first write/erase cycles causes a larger shift in the cell threshold [SKM95] and that this effect will become less noticeable after ten write/erase cycles [HCSL89]. Thus, one should program the FPGA about 100 times with random data, to avoid these effect (suggested for flash/EEPROM memory cells in [Gut01]). The phenomenon of overerasing flash/EEPROM cells can be minimized by first programming all cells before deleting them.

Preventing the Readback Attack: The readback attack can be prevented with the security bits set, as provided by the manufactures, see Section 3.3. If one wants to make sure that an attacker is not able to apply fault injection, the FPGA has to be embedded into a secure environment, where after detection of an interference the whole configuration is deleted or the FPGA is destroyed.

Preventing the Side Channel Attack: In recent years, there has been a lot of work done to prevent side-channel attacks (see, e.g., relevant articles in [KP99,KP00,KNP01,KKP02]). The methods can generally be divide into software and hardware countermeasures, with the majority of proposals dealing with software countermeasures. "Software" countermeasures refer primarily to algorithmic changes, such as masking of secret keys with random values, which are also applicable to implementations in custom hardware or FPGA. Hardware countermeasures often deal either with some form of power trace smoothing or with transistor-level changes of the logic. Neither seem to be easily applicable to FPGAs without support from the manufacturers. However, some proposals such as duplicated architectures might work on today's FPGAs.

5 Open Problems

At this point we would like to provide a list of open questions and problems regarding the security of FPGAs. If answered, such solutions would allow standalong FPGAs with much higher security assurance than currently available.

Side channel attacks Side channel attacks on FPGAs should be investigated with the same intensity as it has been done with processor and ASIC platforms. We assume that many characteristics from ASIC attacks carry over, but some features will certainly be different. One extremely interesting question in this context is whether there are design strategies that make FPGA designs less vulnerable against power analysis attacks. If so, can we integrate such strategies in the design tools?

Fault injection There appears to be no published attempt to perform this kind of attack against FPGAs. It seems very likely that one can use, for example, radiation or glitches to alter the contents of FPGAs. However, for attacks that need to target specific components of a design, the small features of current FPGAs together with the uncertainty of the location of the design might make those fault injection attacks a more formidable task in the FPGA case. We think that this subject is worth further investigation.

Key management for configuration encryption On-chip decryption of an encrypted configuration file would have benefits well beyond cryptographic applications. One major problem to overcome is the key management. The specific problems include who assigns the keys, who keeps track of them, secure storage of the key, and possibly zeroization of the key when an attack is detected.

Secure deletion The configuration time of an FPGA transcends more than one minute and hence it is not possible to overwrite or delete the configuration after an attack was detected. This is bad, especially if the configuration of the FPGA includes any proprietary algorithms which should be kept secret. There should be research on how to securely delete the design information. A related problem is on-chip tamper detection for FPGAs.

Physical attacks There has not been any published physical attacks against FPGAs. Again, we think it would be very interesting to study this problem.

6 Conclusions

This contribution analyzed possible attack against the use of FPGA in security applications. For black box attacks, we stated that they are not feasible for state-of-the-art FPGAs. However, it seems very likely for an attacker to get the secret information stored in a FPGA, when combining readback attacks with fault injection. Cloning of SRAM FPGA and reverse engineering depend on the specifics of the system under attacked, and they will probably involve a lot of effort, but this does not seem entirely impossible. Physical attacks against FPGAs are very complex due to the physical properties of the semiconductors in the case of flash/SRAM/EEPROM FPGAs and the small size of AF cells. It appears that such attacks are even harder than analogous attacks against ASICs. Even though FPGA have different internal structures than ASICs with the same functionality, we believe that side-channel attacks against FPGAs, in particular power-analysis attacks, will be feasible too.

From the discussion above it may appear that FPGAs are currently out of question for security applications. We don't think that this the right conclusion, however. It should be noted that many commercial ASICs with cryptographic functionality are also vulnerable to attacks similar to the ones discussed here. A commonly taken approach to prevent these attacks is to put the ASIC in a secure environment. A secure environment could for instance be a box with tamper sensors which triggers what is called "zeroization" of cryptographic keys, when an attack is being detected. Similar approaches are certainly possible for FPGAs too. (Another solution often taken by industry is not to care and to build cryptographic products with poor physical security, but we are not inclined to recommend this.)

References

[AARR] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side – Channel(s). In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2002*, pages 29–45. Springer-Verlag. LNCS 2523.

[Act02] Actel Corporation. Design Security in Nonvolatile Flash and Antifuse. http://www.actel.com, August 2002.

[AEC99] J. M. Aplan, D. D. Eaton, and A. K. Chan. Security Antifuse that Prevents Readout of some but not other Information from a Programmed Field Programmable Gate Array. United States Patent, No. 5898776, April 27 1999.

[Ajl95] C. Ajluni. Two New Imaging Techniques to Improve IC Defect Indentification. *Electronic Design*, 43(14):37–38, July 1995.

[AK97] R.J. Anderson and M.G. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *5th International Workshop on Security Protocols*, pages 125–136. Springer-Verlag, 1997. LNCS 1361.

[Alg] Algotronix Ltd. Method and Apparatus for Secure Configuration of a Field Programmable Gate Array. PCT Patent Application PCT/GB00/04988.

[Amp] Amphion. High Performance AES Encryption Cores. http://www.chipcenter.com.

[ASH+93] Seiichi Aritome, Riichiro Shirota, Gertjan Hemink, Tetsup Endoh, and Fujio Masuoka. Reliability Issues of Flash Memory Cells. *Proceedings of the IEEE*, 81(5):776–788, May 1993.

[Aus95] K. Austin. Data Security Arrangements for Semicondutor Programmable Devices. United States Patent, No. 5388157, Feburary 7 1995.

[BDL97] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *Advances in Cryptology — EUROCRYPT '97*, pages 37–51. Springer-Verlag, 1997. LNCS 1233.

[BS97] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology — CRYPTO '97*, pages 513–525. Springer-Verlag, 1997. LNCS 1294.

[DA99] T. Dierks and C. Allen. *RFC 2246: The TLS Protocol Version 1.0.* Corporation for National Research Initiatives, Internet Engineering Task Force, Network Working Group, Reston, Virginia, USA, January 1999.

[Dip] B. Dipert. Cunning circuits confound crooks. http://www.e-insite.net/ednmag/contents/images/21df2.pdf.

[Dou99] R. Doud. Hardware Crypto Solutions Boost VPN. *Electronic Engineering Times*, pages 57–64, April 1999.

[Eri99] C. R. Erickson. Configuration Stream Encryption. United States Patent, No. 5970142, October 19 1999.

[EYCP01] A. Elbirt, W. Yip, B. Chetwynd, and C. Paar. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. *IEEE Transactions on VLSI Design*, 9(4):545–557, August 2001.

[Fed77] Federal Information Processing Standards, National Bureau of Standards, U.S. Department of Commerce. *NIST FIPS PUB 46, Data Encryption Standard*, January 15, 1977.

[FKK96] A. O. Freier, P. Karlton, and P. C. Kocher. *The SSL Protocol Version 3.0.* Transport Layer Security Working Group INTERNET-DRAFT, November 1996.

[GC01] K. Gaj and P. Chodowiec. Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays. In D. Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume LNCS 2020, pages 84 – 99, Berlin, April 8-12 2001. Springer-Verlag.

[Gut96] P. Gutmann. Secure Deletion of Data from Magnetic and Solid-State Memory. In *Sixth USENIX Security Symposium*, pages 77–90, July 22-25, 1996.

[Gut01] P. Gutmann. Data Remanence in Semiconductor Devices. In *10th USENIX Security Symposium*, pages 39–54, August 13–17, 2001.

[HCSL89] Sameer Haddad, Chi Chang, Balaji Swaminathan, and Jih Lien. Degradations due to hole trapping in flash memory cells. *IEEE Electron Device Letters*, 10(3):117–119, March 1989.

[Jef02] G. P. Jeffrey. Field programmable gate arrays. United States Patent, No. 6356637, March 12 2002.

[KA98] S. Kent and R. Atkinson. *RFC 2401: Security Architecture for the Internet Protocol.* Corporation for National Research Initiatives, Reston, Virginia, USA, November 1998.

[KB00] S. H. Kelem and J. L. Burnham. System and Method for PLD Bitstram Encryption. United States Patent, No. 6118868, September 12 2000.

[Kea01] T. Kean. Secure Configuration of Field Programmable Gate Arrays. In *International Conference on Field-Programmable Logic and Applications 2001 (FPL 2001)*, pages 142–151. Springer-Verlag, 2001. LNCS 2147.

[Kes] D. Kessner. Copy Protection for SRAM based FPGA Designs. http://www.free-ip.com/copyprotection.html.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume LNCS 1666, pages 388–397. Springer-Verlag, 1999.

[KK99] O. Kommerling and M.G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smartcard Technology (Smartcard '99)*, pages 9–20, May 1999.

[KKP02] B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, editors. *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2002*, volume LNCS 2523, Berlin, Germany, August 13-15, 2002. Springer-Verlag.

[KNP01] Ç. K. Koç, D. Naccache, and C. Paar, editors. *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2001*, volume LNCS 2162, Berlin, Germany, May 13-16, 2001. Springer-Verlag.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[KP99] Ç. K. Koç and C. Paar, editors. *Workshop on Cryptographic Hardware and Embedded Systems — CHES'99*, volume LNCS 1717, Berlin, Germany, August 12-13, 1999. Springer-Verlag.

[KP00] Ç. K. Koç and C. Paar, editors. *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2000*, volume LNCS 1965, Berlin, Germany, August 17-18, 2000. Springer-Verlag.

[Lip] H. Lipmaa. Fast Software Implementations of AES. http://www.tcs.hut.fi/ helger/aes/rijndael.html.

[LM90] X. Lai and J. Massey. A proposal for a new block encryption standard. In *Advances in Cryptology — EUROCRYPT '90*, pages 389–404, Berlin, Germany, May 1990. Springer-Verlag. LNCS 473.

[LMM91] X. Lai, Y. Massey, and S. Murphy. Markov Ciphers and Differential Cryptoanalysis. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume LNCS 547, Berlin, Germany, 1991. Springer-Verlag.

[Mil86] V. Miller. Uses of elliptic curves in cryptography. In H. C. WIlliams, editor, *Advances in Cryptology — CRYPTO '85*, volume LNCS 218, pages 417–426, Berlin, Germany, 1986. Springer-Verlag.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, Florida, USA, 1997.

[PGP+91] C. Papadas, G. Ghibaudo, G. Pananakakis, C. Riva, P. Ghezzi, C. Gounelle, and P. Mortini. Retention characteristics of single-poly EEPROM cells. In *European Symposium on Reliability of Electron Devices, Failure Physics and Analysis*, page 517, October 1991.

[PWF+00] R. C. Pang, J. Wong, S. O. Frake, J. W. Sowards, V. M. Kondapalli, F. E. Goetting, S. M. Trimberger, and K. K. Rao. Nonvolatile/battery-backed key in PLD. United States Patent, No. 6366117, Nov. 28 2000.

[QS01] J.-J. Quisquater and D. Samyde. Electro Magnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. In *International Conference on Research in Smart Cards, E-smart 2001*, pages 200 – 210, Cannes, France, September 2001.

[Ric98] G. Richard. Digital Signature Technology Aids IP Protection. In EETimes - News, 1998. Available at http://www.eetimes.com/news/98/1000news/digital.html.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[SA93] J. Soden and R.E. Anderson. IC failure analysis: techniques and tools for quality and reliability improvement. *Proceedings of the IEEE*, 81(5):703–715, May 1993.

[SA02] S. Skorobogatov and R.J. Anderson. Optical Fault Induction Attacks. In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2002*, pages 2–12. Springer-Verlag, 2002. LNCS 2523.

[Sch96] B. Schneier. *Applied Cryptography*. John Wiley & Sons Inc., New York, New York, USA, 2nd edition, 1996.

[Sch98] D.K. Schroder. *Semiconducor Material and Device Characterization*. John Wiley and Sons, 1998.

[Sea] G. Seamann. FPGA Bitstreams and Open Designs. http://www.opencollector.org/.

[SKM95] K.T. San, C. Kaya, and T.P. Ma. Effects of erase source bias on Flash EPROM device reliability. *IEEE Transactions on Electron Devices*, 42(1):150–159, January 1995.

[SW99] C. Sung and B. I. Wang. Method and Apparatus for Securing Programming Data of Programmable Logic Device. United States Patent, Patent Number 5970142, June 22 1999.

[TABG03] S. Thomas, D. Anthony, T. Berson, and G. Gong. The W7 Stream Cipher Algorithm. Available at http://www.watersprings.org/pub/id/draft-thomas-w7cipher-03.txt, April 2003. Internet Draft.

[TCH93] Jiang Tao, Nathan Cheung, and Chenming Ho. Metal Electromigration Damage Healing Under Bidirectional Current Stress. *IEEE Transactions on Elecron Devices*, 14(12):554–556, December 1993.

[TG99] R. Taylor and S. Goldstein. A high-performance flexible architecture for cryptography. In *Workshop on Cryptographic Hardware and Embedded Systems — CHES '99*, pages 231–245. Springer-Verlag, August 1999. LNCS 1717.

[U.S01] U.S. Department of Commerce/National Institute of Standard and Technology. *FIPS PUB 197, Specification for the Advanced Encryption Standard (AES)*, November 2001.

[vdPK90] J. van der Pol and J. Koomen. Relation between the hot carrier lifetime of transistors and CMOS SRAM products. In *International Reliability Physics Symposium (IRPS 1990)*, page 178, April 1990.

[WKM+96] T.W. Williams, R. Kapur, M.R. Mercer, R.H. Dennard, and W. Maly. IDDQ Testing for High Performance CMOS - The Next Ten Years. In *IEEE European Design and Test Conference (ED&TC'96)*, pages 578–583, 1996.

[Wu99] H. Wu. Low complexity bit-parallel finite field arithmetic using polynomial basis. In *Workshop on Cryptographic Hardware and Embedded Systems — CHES 1999*, pages 280 – 291. Springer-Verlag, 1999. LNCS 1717.

[WWGP00] T. Wollinger, M. Wang, J. Guajardo, and C. Paar. How well are high-end DSPs suited for the AES algorithms? In *The Third Advanced Encryption Standard Candidate Conference*, pages 94–105, New York, New York, USA, April 13–14 2000. National Institute of Standards and Technology.

[Xil] Xilinx Inc. Using Bitstream Encryption. Handbook of the Virtex II Platform. http://www.xilinx.com.

[YN00] Kun-Wah Yip and Tung-Sang Ng. Partial-Encryption Technique for Intellectual Property Protection of FPGA-based Products. *IEEE Transactions on Consumer Electronics*, 46(1):183–190, 2000.