Implementation Efficiency of Cryptography and AES on FPGAs

Amit Kumar, Manoj Kumar, P. Balaramudu

Abstract—**These FPGAs have become dominant part of embedded systems, it is imperative to consider their security as a whole. It provides a state-of-the-art description of security issues on FPGAs, both from the system and implementation perspectives. We discuss the advantages of reconfigurable hardware for cryptographic applications, show potential security problems of FPGAs, and provide a list of open research problems. Moreover, we summarize both public and symmetric-key algorithm implementations on FPGAs.**

IndexTerms— **Security, cryptography, reconfigurable hardware, AES, FPGA.**

I. INTRODUCTION

FPGAs are becoming integral parts of embedded systems and many embedded applications require security mechanisms because of their very nature, it is imperative to consider security on FPGAs as a whole. Applications will be wireless making the need for security and privacy preserving mechanisms genuine. The configurability of FPGAs offers major advantages when using them for cryptographic applications. We need to know the suitability of FPGAs for security applications from a systems point of view and test the resistance of FPGAs to physical or system attacks, which, in practice, pose for a greater danger than algorithmic attacks. The first part of this paper is devoted to studying FPGAs from a systems security perspective. We do this by looking at attacks documented in the literature against FPGAs as well as attacks that have been performed against other hardware platforms and by adapting them and their solutions to FPGAs. We provide a list of open problems regarding system security of FPGAs. The second part of this work makes an attempt to organize the vast literature on FPGA cryptographic algorithm implementation according to the different hard mathematical problems on which the cryptographic primitives are based. In addition, taking performance (both area and throughput) as a measure of different implementations reported in the literature, we make recommendations as to which methods are best suited to implement cryptographic algorithms on FPGAs. The rapid progress in speed and integration density of commercial FPGAs increase advantages of FPGAs for cryptographic applications. Possible attacks are presented with possible counter measures against the attacks. We discuss possible metrics to be used to compare the FPGA algorithm implementations. We also attempt to list open problems regarding security applications in which FPGAs are used.

Manuscript received June, 2017.

***AmitKumar**,P.G. Student, Pune University, SVCET College, (e-mail: amitkumar_123@rediffmail.com).Rajuri, Pune, India,*

***Manoj Kumar**, E&TC Dept., Pune University, SVCET College,(e-mail: manoj1985.111@rediffmail.com), Rajuri, Pune*

***P. Balaramudu**,Pune University, SVCET College, Rajuri, Pune, (e-mail: balram9hossanna@hotmail.com).*

II SYSTEM ADVANTAGES OF FPGAS FOR CRYPTOGRAPHIC APPLICATIONS

In this section, we list potential advantages of reconfigurable hardware (RCHW) in cryptographic applications.

Algorithm Agility: This term refers to the switching of cryptographic algorithms during operation of the targeted application. Widely held of modern security protocols, such as SSL or IPsec, is 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. Eg. IPsec allows among others DES, 3DES, Blowfish, CAST, IDEA, RC4 and RC6 as algorithms, and future extensions are possible. Advantages of algorithm independent protocols are:

a) Ability to delete broken algorithms

b) Choose algorithms according to certain (e.g. personal) preferences

c) Ability to add new algorithms. Algorithm agility is expensive with old-style hardware. FPGAs canbe reprogrammed in real time.

Algorithm Upload: It is perceivable that fielded devices are upgraded with a new encryption algorithm. A reason for this could be that the product has to be compatible to new applications. From a cryptographically point of view, algorithm upload can be necessary because a current algorithm was broken. E.g., Data Encryption Standard (DES) made by Federal Information Processing Standards, a standard expired. A new standard was created Advanced Encryption Standard - AES by U.S. Department of Commerce/National Institute of Standard and Technology and/or that the list of ciphers in an algorithm independent protocol was extended. Assuming there is some kind of (temporary) connection to a net-work such as the Internet, FPGA-equipped encryption devices can upload the new configuration code.

Architecture Efficiency: In certain cases, 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 (e.g., the specific curve of an ECC system), 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 is with fixed keys and operation of constant multiplication which is far more efficient than a

general modular multiplication. Another example taken from asymmetric cryptography is the arithmetic architectures for Galois fields. These architectures tend to be more efficient if the field order and irreducible polynomial are fixed. Squaring in GF (2m) takes m=2 cycles with a general architecture, but only one cycle if the architecture is compiled for a fixed field. Notice that squaring in GFis one of the most common operations when implementing elliptic curve cryptosystems defined over fields of characteristic two. 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 run-time reconfiguration.

Algorithm Modification: There are applications which require modification of standardized cryptographic algorithms 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 where DES is used 25 times in a row and a 12-bit salt modifies the expansion mapping. It is also attractive to customize block cipher such as DES or AES with proprietary S-boxes for certain applications.

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. Modular arithmetic operations include for example exponentiation for RSA and multiplication, squaring, inversion, and addition for elliptic curve cryptosystems (ECC). 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 and Pentium III 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.

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 re-configured 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.

III. SECURITY THREATS and ATTACK 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. Threats:

The most common threat against an implementation of a 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 and harming past communications 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 communications. In other cases, execution of a certain cryptographic operation with a presuming 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.

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 communications. 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.)

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. The cost of the attack, furthermore, rises with the usage of state machines, LFSRs (Linear Feedback Shift Registers), integrated storage, and, if pins can be used, input and output.

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, proprietary algorithm). The readback functionality can be prevented with a security bit. In some FPGA families, more than one bit is used to disable different features, However, an attacker can overcome these counter measures in FPGA with fault injection. Inserting faults, e.g., electro-magnetic radiation, infrared laser or even a flash light. It seems very likely that these attacks can be easily applied to FPGAs. However, Actel Corporation claims that after the programming phase, the cells of FPGAs cannot be read at all. Xilinx JBits allows a simplified and automated access to specific part of the bitstream, resulting in a extra advantage for the attacker who performs a readback attack.

Cloning of SRAM FPGAs: The security implications that arise in a system that uses SRAM FPGAs are obvious, if the configuration data is stored unprotected in the system but

external to the FPGA. In a standard scenario, the configuration data is stored 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.

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 not only that the attacker has to be in possession of the bitstream, but furthermore the bitstream has to be in the clear, meaning it is not encrypted. 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 NEO Cad reverse-engineered a Xilinx FPGA. NEO Cad was able to reconstruct the necessary information about look-up tables, connections, and storage elements. Hence, NEO Cad 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.

Physical attack: The aim of a physical attack is to investigate the chip design in order to get in-formation 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 : 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. 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. The physical changes are caused mainly by three effects: electro-migration, hot carriers, and ionic contamination. 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 could 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. Hence, the key was "burned" into the memory cells and the key values were retained even after switching on the device. "IDDQ testing" is one of the widely used methods and it is based on the analysis of the current usage of the device. 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 IDDQ characteristic. Access to internal portions of a device, possibilities are to use the scan path that the IC manufacturers insert for test purposes or techniques like bond pad probing. 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 is the traditional way to discover the needed information. Focused Ion Beam (FIB) workstations can expose buried conductors and deposit new probe points. 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 number of secondary electrons that are rejected. 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 transferred. 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 anti-fuse (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 1m2) 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. 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 the configuration file of an Actel A54SX16 chip with 24,000 system gates. Another aggravation of the attack is that only about 2.5 % of all possible connections in an average design are actually used. 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. On the other hand, we know that AF FPGAs can be easily attacked if not connected to a power source. Hence, it is easier to drill holes to disconnect two connections or to

repair destroyed layers. Also, depending on the source, the estimated cost of an attack and its complexity are lower.

Flash FPGAs. The connections in flash FPGAs are realized through flash transistors. That means the number of electrons flowing through the gate changes after configuration and there are no optical differences as in the case of AF FPGAs. Thus, physical attacks performed via analysis of the FPGA cell material are not possible. However, 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. However, experts are not certain about the complexity of such an attack and there is some controversy regarding its practicality. 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. Furthermore, there are long term retention issues, like electron emission. The electrons in the floating gate migrate to the interface with the underlying oxide from where they tunnel into the substrate. This emission causes a net charge loss. The opposite occurs with erased cells where electrons are injected Ionic contamination takes place as well but the influence on the physical behavior is so small that it cannot be measured. 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 [Haddad et al. 1989]. Another phenomenon is over-erasing, where an erase cycle is applied to an already-erased cell leaving the floating gate positively charged. Thus, turning the memory transistor into a depletion-mode transistor. All the described effects change in a more or less extensive way the cell threshold voltage, gate voltage, or the characteristic of the cell.

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. 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. 60% of the power consumption in a XILINX Virtex FPGA is due to the interconnects and 14% and 16% is due to clocking and logic, respectively. These figures would seem to imply that and SPA type attack would be harder to implement on an FPGA.

IV. PREVENTING THE 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. 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. Today's stream ciphers output a bit stream, with a period length of 128 bits [Thomas et al. 2003]. 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 ECC 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 280 possibilities. Hence, implementations of this algorithms cannot 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 will increase complexity as the whole chip, including the serial number can be easily copied and every board would need a different configuration. Another solution would be to use dongles to protect the design. Dongles do not provide solid security, as it can be seen from the software industry's experience using dongles for their tools. 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 counter-measure 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.If an attacker copies the partly decrypted file, the non-decrypted functionality is avail-able, whereas the one decrypted is not. Thus, the attacker tries to find the errors in the design not aware of the fact, that they are caused through the encrypted part of the configuration. Most likely an attacker with little resources, would have dropped the reverse engineering effort, when realizing that parts are decrypted. However, this approach adds hardly any extra complexity to an attack if we assume that an attacker has a lot of resources. One method is different parts of the configuration file are encrypted 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. To overcome this more than one key is stored in the FPGA.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 where only the two keys are stored in the battery powered memory. Due to the fact that the battery powers only a very small memory cells, the battery is limited only by its own life span.

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 cannot 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. Furthermore, the recovery process is heavily dependent on the temperature. 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 uses always the same circuits to feed the secret key to the arithmetic unit. Neutralization of this effect can be achieved by applying an opposite current or by inserting dummy cycles into the circuit .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. In terms of flash/EEPROM memory cell, one has to consider that the first write/erase cycles cause a larger shift in the cell threshold and that this effect will become less noticeably after ten write/erase cycles. Thus, one should program the FPGA about 100 times with random data, to avoid these effects. The phenomenon of over erasing flash/EEPROM cells can be minimized by first programming all cells before deleting them.

Preventing the ReadbackAttack :The readback attack can be prevented with the security bits set, as provided by the manufactures. 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 :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.

V. AES ON FPGA

Multiple FPGA implementation studies have been presented for the AES candidate algorithm finalists the results of which are discussed below for feedback modes of operation. A similar exercise can beperformed for non-feedback modes. The studies performed byElbirt et al. used a Xilinx Virtex XCV1000-4 as the target FPGA. The studyperformed by Dandalis et al. used the Xilinx Virtex Family but did notspecify which FPGA was used as the target device, this makes comparison withother implementations very hard.PP-2, LU-1, and LU-8, correspond to partial pipelining with twostages, one round loop unrolling, and eight rounds loop unrolling architectures,respectively. In addition, notice that the implementation of a one stage partialpipeline, an iterative looping architecture, a one round loop unrolled architectureimplementations which used the Xilinx Virtex XCV1000 as their hardware platformbecause it is the most common platform among published works, thus making acomparison somewhat reasonable. One can see that most implementations achievesimilar throughputs for the same algorithms. In addition, if one were to choosealgorithms based on their throughput all authors would agree that Serpent wouldwin followed closely by Rijndael, Twofish and RC6, and MARS at the end4. Themost interesting effect is that Gaj and Chodowiec achieved similar performanceas other implementations at the cost of half the area, which is due, according to theauthors, to resource sharing. If we were to perform a similar exercise in terms of theTPS ratio, then Rijndael would win followed by Serpent and Twofish, and finally,RC6 and MARS. One reason for RC6 and MARS to have the poorest performancewhen implemented on FPGAs is their use of a multiplier in their round function.We refer to Gaj and Chodowiec ,Elbirt et al. ,Nechvatal et al formore in depth architecture and performance comparisons of the AES candidates.FPGA implementations of individual candidate algorithms (both finalists andnon-finalists) have also been performed. Implementations of CAST-256 achievedthroughputs of 11.03 Mbits/sec using a Xilinx Virtex XCV1000-4and13 Mbits/sec using a Xilinx XC4020XV-9. An RC6 implementation achieved a throughput of 37 Mbits/sec using a Xilinx XC402OXV-9. A Serpent implementation using a Xilinx Virtex XCV1000-4achieved a throughput of 4.86 Gbits/sec .When targeted toa Xilinx Virtex-E XCV400E-8, a Serpent implementation achieved a throughputof 17.55 Gbits/sec through the use of the Xilinx run-time reconfiguration softwareapplication JBitsTM which allowed for real time key-specific compilation of thebit-stream used to program the FPGA [Patterson 2000a]. This run-time reconfiguration resulted in a smaller and faster design (which operated at 137.15 MHz)as compared to the design in of

Elbirt and Paar(which operated at 37.97MHz). When implemented using an FPGA from the Altera Flex 10KA Family, theSerpent algorithm achieved a maximum throughput of 301 Mbits/sec however, it is important to note that this implementation implements 8 of the Serpent's algorithm thirty two rounds whilethe implementations by Elbirt and PaarandPatterson implement allthe rounds of the Serpent algorithm. Note that all of the presented throughputvalues are for non-feedback modes of operation.Multiple implementations of Rijndael, the AES, have been presented using bothXilinx and Altera FPGAs.The implementation in by McLoone and McCannyachieves a throughputof 6.956 Gbits/sec using a Xilinx Virtex-E XCV3200E-8. Utilizing ROM to implement the Rijndael Byte-Sub operation resulted in a significant increase in through-put and decrease in area as compared to implementations in Dandalisat theexpense of BRAM Blocks which the previous implementations did not use. Whentargeting the more advanced Altera APEX20KE200-1, Rijndael implementationsachieved throughputs ranging from 570 Mbits/sec to 964 Mbits/sec. depending on the implementation methodology. Four recent implementations are also worth mentioning. The implementations achieve throughput rates of11.8 and 17.8 Gbits/s, respectively. These throughputs are only achieved throughpipelining and thus are not suitable for feedback modes of operation. Notice that JÄarvinen achieve such high throughputs for a 128-bit key Rijndael implementation. The work presented in Standaert et al. is also interesting because it compares different implementation options (Look-up table basedimplementations, RAM-based implementations, and composite field implementations) and proposed some heuristics to evaluate the hardware efficiency at differentsteps of the design process which result on particularly efficient implementations.Finally, we include the work presented in Chodowiec and Gaj as its targetis the implementation of a resource efficient AES core on FPGAs. Such work hasnot been considered extensively in the literature (usually designs are optimized forspeed rather than area when targeting FPGAs). In addition, they proposed a newway of implementing the MixColumns and InvMixColumns transformations whichreduces area and might be interesting in its own right. This work achieves data streams of 150 Mbits/sec for encryption and decryption ona low-cost Xilinx Spartan II FPGA using 222 slices and 3 BRAMs.

V CONCLUSION

We analyzed possible attacks against the use of FPGA in security applications. Black box attacks do not seem to be 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 and fault injection attacks. Cloning of SRAM FPGA and reverse engineering depend on the specifics of the system under attack, 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.AES have been thoroughly investigated and different approaches to implementations exposed in the context of embedded systems.

While, the art of cryptographic algorithm implementation is reaching maturity, FPGAs as a security platform are hot topic for security applications. FPGA in a secure environment, for instance, be a box with tamper sensors which triggers what is called ‘zeroization’ of cryptographic keys, when an attack is being detected.

REFERENCES

[1] Actel Corporation. 2002. Design Security in Nonvolatile Flash and Antifuse. Available at http://www.actel.com.

[2] Agrawal, D., Archambeault, B., Rao, J. R., and Rohatgi P.2002. The EM Side Channel(s) In Workshop on CryptographicHardware and Embedded Systems.

[3] Ajluni, C. 1995. Two New Imaging Techniques to Improve ICDefect Identification. Electronic Design 43, 14 (July), 3738.

[4] Algotronix Ltd. Method and Apparatus for Secure Configuration of a Field Programmable Gate Array PCTPatent Application 04988.

[5] Altera Corporation 2000. Nios Soft Core Embedded Processor Altera Corporation.

[6] Altera Corporation 2002a. Excalibur Device Overview, Altera Corporation. Av. http://www.altera.com.

[7] Altera Corporation 2002b. Stratix FPGA Family,Altera Corporation.Av.at http://www.altera.com.

[8] American National Standards Institute 1998. ANSI X9.52-998,Triple Data Encryption Algorithm Modes ofOperation.American National Standards Institute. http://webstore.ansi.org/.

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

[10] Anderson, R. and Kuhn, M. 1997. Low Cost Attacks OnTamper Resistant Devices.In 5thInternational Workshop onSecurity Protocols,B. Christianson, B. Crispo, T. M. A. Lomas and M. Roe, Eds. Vol. LNCS

[11] ANSI. 1981. American National Standards Data E ncryptionAlgorithm X3.92-1981. American National Standards Association.

[12] Aplan, J. M., Eaton, D. D., and Chan, A. K. 1999 SecurityAntifuse that Prevents Readout of some but not otherInformation from a Programmed Field Programmable GateArray.United States Patent, Patent Number 5898776.

Amit Kumar,ME Student, Sahyadri Valley College of Engineering, Pune University.
Manoj Kumar,Head, E&TC Dept., Sahyadri Valley College of Engineering, Pune University.
P. Balramudu, Sahyadri Valley College of Engineering, Pune University.