Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker

Sandeep Kumar[1], Christof Paar[1], Jan Pelzl[1], Gerd Pfeiffer[2], and Manfred Schimmler[2]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{kumar,cpaar,pelzl}@crypto.rub.de
[2] Institute of Computer Science and Applied Mathematics, Faculty of Engineering, Christian-Albrechts-University of Kiel, Germany
{gp,masch}@informatik.uni-kiel.de

Abstract. Cryptanalysis of symmetric and asymmetric ciphers is computationally extremely demanding. Since the security parameters (in particular the key length) of almost all practical crypto algorithms are chosen such that attacks with conventional computers are computationally infeasible, the only promising way to tackle existing ciphers (assuming no mathematical breakthrough) is to build special-purpose hardware. Dedicating those machines to the task of cryptanalysis holds the promise of a dramatically improved cost-performance ratio so that breaking of commercial ciphers comes within reach.
This contribution presents the design and realization of the COPACOBANA (Cost-Optimized Parallel Code Breaker) machine, which is optimized for running cryptanalytical algorithms and can be realized for less than US$ 10,000. It will be shown that, depending on the actual algorithm, the architecture can outperform conventional computers by several orders in magnitude. COPACOBANA hosts 120 low-cost FPGAs and is able to, e.g., perform an exhaustive key search of the Data Encryption Standard (DES) in less than nine days on average. As a real-world application, our architecture can be used to attack machine readable travel documents (ePass). COPACOBANA is intended, but not necessarily restricted to solving problems related to cryptanalysis.
The hardware architecture is suitable for computational problems which are parallelizable and have low communication requirements. The hardware can be used, e.g., to attack elliptic curve cryptosystems and to factor numbers. Even though breaking full-size RSA (1024 bit or more) or elliptic curves (ECC with 160 bit or more) is out of reach with COPACOBANA, it can be used to analyze cryptosystems with a (deliberately chosen) small bitlength to provide reliable security estimates of RSA and ECC by extrapolation[3].

1 Introduction

All modern practical ciphers, both symmetric and asymmetric ones, use security parameters (in particular the key-length) which makes them secure against attacks with current computers. Depending on the security margin chosen in a given application, many ciphers are potentially vulnerable to attacks with special-purpose machines which have, say, a cost-performance ratio which is several orders of magnitude better than that of current PCs. This contribution describes a design and successful prototype realization of such a special-purpose cryptanalytical machine based on low-cost FPGAs.

Cryptanalysis of modern cryptographic algorithms requires massive computational effort, often between 2^{56} to 2^{80} operations. A characteristic of many (but not all) cryptanalytical algorithms is that they can run in a highly parallel fashion with very little interprocess communication. Such applications map naturally to a hardware based design, requiring repetitive mapping of the basic block, and can be easily extended by adding more chips as required. However, it should be stressed that the mere availability of computational resources is not the

[3] The basic architecture of COPACOBANA was presented as a poster at a hardware workshop (not disclosed here in order to keep this submission anonymous).

core problem, but providing massive computational resources *at affordable costs* is. The non recurring engineering costs for ASICs have put special-purpose hardware for cryptanalysis in almost all practical situations out of reach for commercial or research institutions, and have been considered only feasible by government agencies.

An alternative approach to distributed computing with loosely coupled processors is based on using the idle cycles of the huge number of computers connected via the Internet, for instance the SETI@home project [16]. The results of this approach has been quite successful for some applications (even though the confirmed detection of extraterrestrial life is still an open problem) and is used for selected problems which are not viable with the computing power within a single organization. Using distributed computing, however, has the disadvantage of, first, having to find individuals who would be interested in joining to solve a problem and, secondly, trusting the nodes from introducing errors. Finally, for many code-breaking application, shared computation is not a method of choice in many cases.

With the recent advent of low-cost FPGA families with much logic resources, field programmable gate arrays provide a very interesting alternative tool for the massive computational effort required for cryptanalytic applications. Reconfigurable computing has been emerged as a cost effective alternative for various applications which require the power of a custom hardware but require the flexibility provided by a software based design, e.g., in rapid prototyping. In addition, to the cost-performance advantage over PC-based machines, such a machine has the advantage over ASIC-based designs that it can be used to attack various different cryptosystems without the need to rebuilt a new machine each time. In cryptanalysis, certain algorithms are very well suited for special-purpose hardware. A prime example for this is an exhaustive key search of the Data Encryption Standard (DES) [10]. Such a brute-force attack is more than two orders of magnitude faster when implemented on FPGAs than in software on general purpose computers at equivalent costs[4].

This contribution describes the design, implementation, and applications of COPACOBANA, a massively parallel machine based on FPGAs. The hardware is suitable for computational problems which are parallelizable and have low communication requirements and can be used, e.g., to attack elliptic curve cryptosystems and to factor numbers. Even though breaking full-size RSA (1024 bit or more) or elliptic curves (ECC with 160 bit or more) is out of reach with COPACOBANA, it provides for the first time a tool for a reliable security estimation of RSA and ECC. Even more relevant is the fact that resource constrained applications, in particular mobile devices, sometimes settle with shorter parameters, such as the 112 bit and 128 bit ECC systems recommended by the SECG standard, which become vulnerable with our machine. Also, assuming Moore's law, we can predict the security margin of RSA and ECC in the years to come.

Another interesting application emerges in the area of machine readable travel documents (ePass): The International Civil Aviation Organization (ICAO) initiated biometric and RFID technologies for border and visa control. Current realizations of Basic Access Control deploy symmetric cryptography (Triple-DES) and generate the corresponding encryption and authentication keys from passport information. As pointed out by many experts however, the low entropy of the key allows for attacks of complexity of not more than single DES. Using our hardware architecture this kind of attack can be mounted in much shorter time, and even real-time, i.e., the time needed to pass the inspection system.

[4] Based on our existing implementations, a single FPGA at a cost of US$ 50 (current market price) can test 400 million keys, a PC (Pentium4, 2GHz) for US$+ 200 approx. 2 million keys per second. Hence, 4 FPGAs can perform the same task approximately 800 times faster than a PC at the same cost.

The outline of the paper is as follows: In the next Section, we identify a model for an optimized hardware architecture for breaking codes which we realized as a custom-designed computing machine. We will present the architectural concept and the prototype of COPACOBANA, consisting of a backplane, an FPGA DIMM module, and a controller card. In Section 3, cryptanalytical applications which are suited for running on low-cost FPGAs will be discussed: First, we show how cryptographically weak systems can be attacked with COPACOBANA. An implementation of the Data Encryption Standard (DES) on COPACOBANA impressively shows how DES can be broken with low effort in less than nine days, making many existing legacy implementations of DES vulnerable to attacks by nearly everyone. Furthermore, we show how the DES implementation at hand can be used for attacks on machine readable travel documents, which use Triple-DES with keys of low entropy. Secondly, we briefly sketch how an efficient hardware implementation of the elliptic curve method (ECM) on COPACOBANA can be used to factor composite integers in parallel. As another asymmtetric cryptanalytical example, a specially tweaked implementation of Pollard's rho algorithm, can be used for breaking elliptic curve cryptosystems (ECC).

2 Proposed Architecture for Cryptanalysis

As we will see in Section 3, many algorithms tackling the most important problems in cryptanalysis can be implemented on FPGAs. However, code breaking involves more effort than programming just a single FPGA with a particular algorithm. Due to the enormous dimensions of cryptanalytical problems, much more resources than a single FPGA are required. What is needed is a powerful massively parallel machine, tweaked to the needs of the targeted algorithms.

Most problems can be parallelized and are perfectly suited for a distributed architecture. In many cases, not much communication overhead is required. Conventional parallel computing architectures, such as provided by Cray, can in theory also be used for cryptanalytical applications. However, the cost-performance ratio is not optimized with this approach, resulting in prohibitively expensive attack machines. Similarly, many features of current high-end processors are not required for the targeted cryptanalytical problems. For instance, high-speed communication between CPUs, fast floating point operations, etc., cannot be used in our context. All of these features usually increase the cost of such a device, which is in particular annoying when they are superfluous. Even a simple grid of conventional PCs is not efficient, as can be seen from implementations of DES: An implementation on a single FPGA can be more than 100 times faster than an implementation on a conventional PC, while the FPGA is much cheaper than the PC. Therefore, a custom design is inevitable in order to obtain a low-cost architecture with the required performance.

Our metric to decide whether an architecture is "good" or not is a function of performance, flexibility, and monetary cost. A good performance metric for hardware implementations is the area-time (AT) complexity. Whenever we can minimize the AT-complexity, the design can be called efficient. ASIC implementations can be AT-minimal and are the best choice for high-volume applications. However, ASICs are not flexible since they can implement only a single architecture. FPGAs in contrast are reprogrammable and, thus, are flexible. Moreover, if only a relatively small number of chips ($<$ 10 000) is required, FPGAs are preferable since the production of ASICs is profitable only when targeting high volumes.

In the following, we describe an optimized architecture for cryptanalytical purposes and its implementation as custom-designed FPGA machine which hosts 120 FPGAs and can be produced for less than US$ 10,000, including material and manufacturing costs.

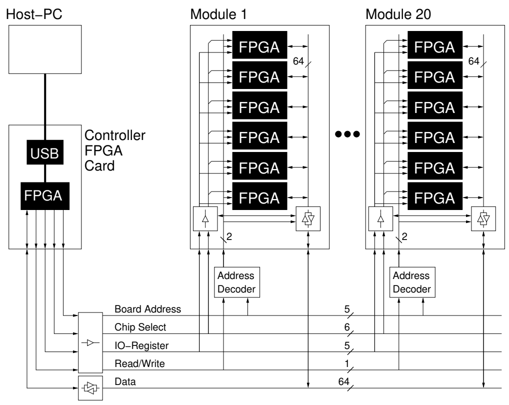
2.1 An Optimal Architecture to Break Ciphers

All targeted algorithms (see Section 3) have the following common characteristics: First, the computational expensive operations are parallelizable. Secondly, single parallel instances do not need to communicate with each other. Thirdly, the overall communication overhead is low, driven by the fact that the computation phase heavily outweighs the data input and output phases. In fact, computation time dominates compared to the time for data input or output. Ideally, communication is almost exclusively used for initialization and reporting of results. A central control instance for the communication can easily be accomplished by a conventional (low-cost) PC, connected to the instances by a simple interface. No high-speed communication interface is required. Forthly, all presented algorithms and their corresponding implementations call for very little memory. As a consequence, the available memory on contemporary low-cost FPGAs such as the Xilinx Spartan3 is sufficient.

2.2 Realization of COPACOBANA

Recapitulating, the Cost-Optimized Parallel Code Breaker (COPACOBANA) fitting our needs consists of many independent low-cost FPGAs, connected to a host-PC via a standard interface, e.g., USB or Ethernet. Furthermore, such a standard interface allows to easily extend a host-PC with more than one COPACOBANA device. The initialization of FPGAs, the control, and the accumulation of results is done by the host. Since the cryptanalytical applications demand for plenty of computing power, the targeted platform aggregates up to 120 FPGAs (Spartan3-1000). Building a system of such a dimension with commercially available FPGA boards is certainly feasible, but comes with a cost penality. Hence we decided to design, layout, and build our own hardware. We considered several different design options. Our cost-performance optimized design became only feasible by strictly restricting all functionality to those directly necessary for code breaking, and to make several design choices based on readily available components and interfaces. The design of COPACOBANA is depicted in Figure 1 and consists of

Fig. 1. Architecture of COPACOBANA

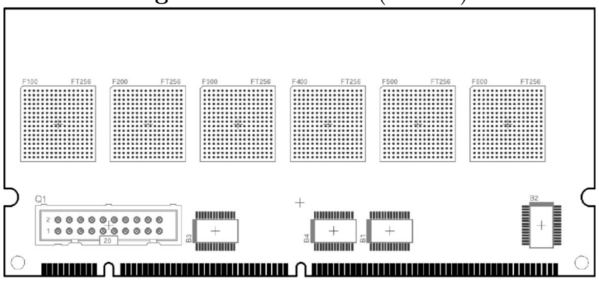


- *FPGA modules* for the actual implementation of the presented hardware architectures,
- a *backplane*, connecting all FPGA modules to a common data bus, address bus, and power supply,
- and a *controller card*, connecting the data bus and address bus to a host-PC via USB.

FPGA Modules: We decided to pick a contemporary low-cost FPGA for the design, the Xilinx Spartan3-1000 FPGA (XC3S1000, speed grade -4, FT256 packaging). This comes with 1 million system gates, 17280 equivalent logic cells, 1920 Configurable Logic Blocks (CLBs) equivalent to 7680 slices, 120 Kbit Distributed RAM (DRAM), 432 Kbit Block RAM (BRAM), and 4 digital clock managers (DCMs) [20].The choice for this chip was derived by an evaluation of size and cost over several FPGA series and types.

A step towards an extendable and simple architecture has been accomplished by the design of small pluggable FPGA modules. We decided to settle with small modules in the standard DIMM format, comprising 6 Xilinx XC3S1000 FPGAs. Figure 4 (Appendix A) shows its realization as custom made 4-layer printed circuit board. The FPGAs are directly connected to a common 64-bit data bus on board of the FPGA module which is interfaced to the backplane data bus via transceivers with 3-state outputs. While disconnected from the bus, the FPGAs can communicate locally via the internal 64-bit bus on the DIMM module. The DIMM format allows for a very

Fig. 2. FPGA module (DIMM)



compact component layout, which is important to closely connect the modules by a bus. Figure 2 depicts the chip arrangement. From the experience with current implementations on the same FPGA type, we dispense with active cooling of the FPGAs at these times. Depending on the heat dissipation of future applications, passive or active cooling might be an option for an upgrade.

Backplane: The backplane hosts all FPGA-modules and the controller card. All modules are connected by a 64-bit data bus and a 16-bit address bus. This single master bus is easy to control because no arbiter is required. Interrupt handling is totally avoided in order to keep the design as simple as possible. If the communication scheduling of an application is unknown in advance, the bus master will need to poll the FPGAs.

Moreover, the power supply is routed to every FPGA module and the controller interface. The backplane distributes two clock signals from the controller card to the slots. Every FPGA module is assigned a unique hardware address, which is accomplished by Generic Array Logic (GAL) attached to every DIMM socket. Hence, all FPGA cores can have the same configuration and all FPGA modules can have the same layout. They can easily be replaced in case of a defect. Figure 5 (Appendix A) shows the prototype of the backplane equipped with one FPGA module and the control interface card which will be described in the next subsection. The entire bus has been successfully tested by use of the prototype FPGA module with frequencies of up to 50 MHz. For the fully equipped board, the bus speed will be limited to 33 MHz due to power dissipation.

Control Interface: Data transfer from and to the FPGAs and to the host-PC is accomplished by the control interface. We decided to pick a small development board with an FPGA (CESYS USB2FPGA [3]) in favor of a flexible design. The development board comes with a Xilinx XC2S200 SPARTAN II FPGA (PQ208), an integrated USB controller (CYPRESS FX-2), and 1 MByte SRAM. Moreover, the board provides an easy-pluggable 96-pin connector which we use for the connection to the backplane. In later versions of the design, it is also possible to replace the FPGA development board by a small microcontroller with a standard USB or Ethernet interface.

The controller hardware has to handle the adaptation of different clock rates: The USB interface uses a clock rate of 24 MHz, the backplane is clocked with 33 MHz, and the controller itself is running at an internal clock of 133 MHz. The internal clock is generated by an external clock synthesizer, the system clock is derived from a digital clock manager (DCM) present on the FPGA.

The main state machine of the control interface is used to decode and execute host commands received via USB, program the FPGAs via the data bus in slave parallel mode, initialize (write to) FPGAs and start the computation, and regularly poll the FPGAs and check for new results.

Programming can be done for all FPGAs simultaneously, for a set of such, or for a particular one. Since the targeted cryptanalytic applications do not require different code on distinct FPGAs, a concurrent programming of all devices is very helpful.

Host-PC: The top level entity of COPACOBANA is a host-PC which is used to program and control all FPGA implementations. For this purpose, a software library has been written to issue commands to the USB connected controller card of COPACOBANA. All software routines are based on the closed source library provided by the board manufacturer (CESYS). With the low-level functions, FPGAs can be addressed and data can be stored and read to/ from a particular FPGA. Further functions include the detection of the hardware and some configuration routines such as, e.g., a backplane reset. Higher-level functions comprise commands at application level. E.g., for the DES Cracker, we can store a certain plaintext in the DES units, check its status, etc.

3 Cryptanalytic Motivation for COPACOBANA

In this section, we will point to possible applications in cryptanalysis. COPACOBANA can be used to break cryptographically weak or outdated algorithms such as DES, A5, and SHA-1 which have an attack complexity of at most 2^{70} operations. But, clearly, COPACOBANA can not recover keys from actual strong cryptosystems such as AES, ECC, and RSA. However, the hardware approach allows to implement attacks on such systems with a deliberately chosen small bitlength and to extrapolate the results to finally obtain a much better estimate of the security of actual cryptosystems against attacks with special-purpose hardware.

We will investigate the complexity of following attacks:

- An exhaustive key search of DES (Subsection 3.1). DES still is used for compatibility reasons and/ or in legacy products. Out-dated DES-based cryptosystems such as Norton Diskreet (a very popular encryption tool in the 1990ies which was of the well-known Norton Utilities package) can be broken with COPACOBANA. Diskreet was used to encrypt single files as well as to create and manage encrypted virtual disks.

- Attacks on machine readable travel documents (ePass): With the DES implementation at hand, an intimidating real-world example of a weak cryptosystem, namely the recently introduced ePass by ICAO, can be attacked in certain ways which we will sketch in Subsection 3.2.
- Factoring composites with the elliptic curve factorization method (ECM) (Subsection 3.3). ECM can be used as a crucial step for factoring actual RSA moduli and a reliable estimate of its complexity is indispensable for the security evaluation of factorization-based cryptosystems such as RSA.
- Attacks against ECC with a parallel variant of Pollard's rho method (Subsection 3.4). The hardware implementation of an algorithm solving the discrete logarithm problem on elliptic curves gives rise to a more realistic estimate of the security of ECC against attacks with special-purpose hardware.

3.1 Exhaustive Key Search of DES

Ideally, the security of symmetric ciphers is dependent on the impracticability of an exhaustive key search. This requires examining through each key in the possible key space. The cost of the attack is calculated based on the available technology and expected future developments. Usually, the key size is chosen such that it allows for a fast and efficient implementation of the cryptosystem but making such brute force attacks impracticable.

The Data Encryption Standard (DES) with a 56-bit key size was chosen as the first commercial cryptographic standard by NIST in 1977 [10]. A key size of 56-bits was considered to be good choice considering the huge development costs for computing power in the late 70's, making a search over all the possible 2^{56} keys impractical. But DES has survived long beyond its recommended lifetime and still is being used in legacy systems or due to backward compatibility reasons. The advances in the hardware and decreasing costs have made DES vulnerable to brute force attacks.

Previous Work: There has been a lot of feasibility studies on the possible use of parallel hardware and distributed computing for breaking DES. The first estimates were proposed by Diffie and Hellman [5] for a brute force machine that could find the key within a day at US$ 20 million.

A first ever detailed hardware design description for a brute force attacker was presented by Michael Wiener at the rump session of CRYPTO'93 and is reprinted in [18]. The machine could be built for less than a million US$ with $57,000$ DES chips that could recover a key every three and half hours. The estimates were updated in 1998 due to the advances in hardware for a million dollar machine to 35 minutes for each key recovery [19].

Ian Goldberg and David Wagner estimated the cost for building a DES brute force attacker using FPGAs at US$ 45,000 for a key recovery within a year [6]. In 1997, a detailed cost estimate for three different approaches for DES key search: distributed computing, FPGAs and custom ASIC designs, was compiled by a group of cryptographers [1].

The real practical attempts at breaking DES were encouraged by the RSA Secret Key challenge launched in 1997 [15]. The first challenge was solved by Rocke Verser, Matt Curtin, and Justin Dolske using the DESCHALL distributed network in 1997. The RSA DES Challenge II-1 was broken by *distributed.net* within 39 days in 1998. The RSA DES Challenge II-2 was won by the Electronic Frontier Foundation (EFF) DES hardware cracker called *Deep Crack* in 1998 within 56 hours [6]. The DES cracker consisted of $1,536$ custom designed ASIC chips at a cost of material of around US$ 250,000 and could search 88 billion keys per second. The final blow to DES was given by the DES Challenge III which was solved in 22 hours 15 minutes using the combined effort of *Deep Crack* and *distributed.net*

A first low-cost approach in attacking a DES-based protocol was realized by [4]. The authors describe their experiences attacking the IBM 4758 CCA with an off-the-shelf FPGA development board.

Though this proved to be an end for DES for many applications, the huge cost involved to producing a machine like *Deep Crack* and access to foundries makes building such machines still impractical for smaller organizations. Therefore, we propose a more practical approach of an off-the-shelf-FPGA based hardware cracker.

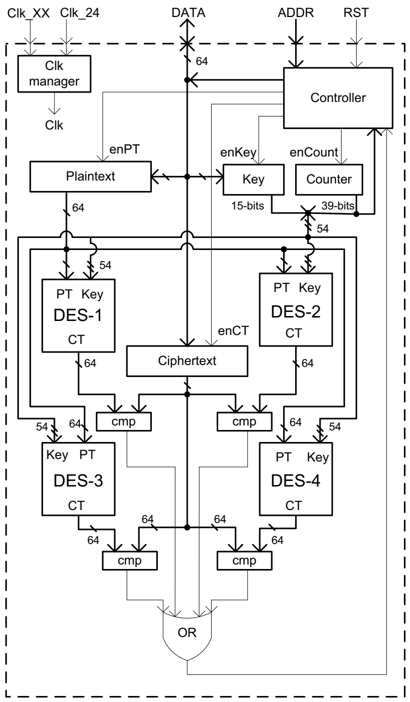
DES on FPGAs: When DES was first proposed as a standard, its main application was seen in hardware based implementations. Hence DES is extremely efficient in terms of area and speed for hardware but unsuitable for a good software implementation due to the bit-level addressing in the design. Therefore an FPGA implementation of DES can be more than a 100 times faster than an implementation on a conventional PC at much lower costs. This allows a hardware based key search engine to be much faster and efficient compared to a software based approach.

The main aim of our key search engine is to check as many keys as possible in the least time to find the right key that could encrypt a known plaintext to its ciphertext that is made available. It is obvious that such a key search can be done in a highly parallelized fashion by partitioning the key space. This requires hardly any inter-process communication, as each of the DES engines can search for the right key within its allocated key subspace.

For the DES engine, we implemented a highly pipelined design of the Université Catholique de Louvain's Crypto Group [14]. The design can test one key per clock per engine and the pipelined architecture is adjusted such that the critical path is as small as possible, allowing for a fast implementation. For COPACOBANA, we can fit four such DES engines inside a single FPGA, and therefore allow for sharing of control circuitry and the key space as shown in Figure 3. It consists of a 64-bit *Plaintext* register and 64-bit *Ciphertext* register. The key space is allocated to each chip as the most-significant 15-bits of the key which is stored in the *Key* register. The *Counter* is used to run through the least significant 39 bits of the key. The remaining two bits of the 56-bit key for each of the DES engines is hardwired and is different for each of them. Thus, for every such FPGA, a task is assigned to search through all the keys with the 15 most-significant bits fixed, that is 2^{41} different keys. The partitioning of the key space is done by the host-PC such way that each chip takes around 90 minutes (at 100 MHz) to check through its allocated key subspace, thus, avoiding huge communication requirements. This also allows the machine to restart the key search easily from a previous state if a power failure occurs. The generated cipher text (*CT*) is compared to that of the given *Ciphertext* stored in the register, using the comparator (*cmp*) block. The results of the four comparators are ORed and reported to the controller. If any of the DES engines provides a positive match, the controller reports the counter value to the host-PC. The host-PC keeps track of the key range that is assigned to each of the FPGAs and, hence, can match the right key from a given counter value. If no match is found until the counter overflows, the FPGA reports completion of the task and remains idle until a new key space is assigned. Since each FPGA can search through its key space totally independent of any other FPGA, only the host-PC needs to keep track of the number of FPGAs and the allocated key space. The estimated time to complete the key search using COPACOBANA is discussed in the following.

Exhaustive Key Search with COPACOBANA: We can operate each of the FPGAs at 100 MHz and, therefore, each FPGA can check four keys every 10 ns. Consequently, a partial key space of 2^{41} keys can completely be checked in $2^{39} \cdot 10$ ns by a single FPGA, which is approximately 92 minutes. Since COPACOBANA hosts 120 of these low-cost FPGAs, the key

Fig. 3. Overview of an FPGA with four DES key search units



search machine can check $4 \cdot 120 = 480$ keys every 10 ns, i.e., 48 billion keys per second. To find the right key, COPACOBANA has to search through an average of 2^{55} different keys. Thus on average, COPACOBANA can find the right key after $(2^{55} \cdot 10)/480$ ns which is approximately 8.7 days. The time required for loading the plaintext, ciphertext and key space allocation are ignored as they are negligibly small compared to the overall running time.

3.2 ePass

One important application of our architecture concerns the current scheme for machine readable travel documents, also known as ePass, which is initiated by organizations[5] in United States and several other countries to deploy biometric and RFID technologies for border and visa control. The claimed goal is to enhance security, protect against forgery and manipulation of travel documents and ease identity checks. The initiative has been subject to many political and technical debates. Several researchers have pointed out the security and privacy weaknesses of the deployed schemes and proposed improvements (see, e.g. [8, 9]). The cryptographic parts of the scheme shall consist of a Passive Authentication, Basic Access Control and an Active Authentication. Whereas Passive Authentication means that the data stored on an ePass are signed by the issuing nation, Basic Access Control should setup a secure (confidential) channel between the reader device (part of the inspection system) and the ePass chip and Active Authentication is deployed for anti-clonig purposes and requires an integer factorization based signature scheme implemented on the ePass chip. Note, that both Basic Access Control and Active Authentication are optional mechanisms. Basic Access Control is already implemented, e.g., in Germany and the Netherlands.

[5] More concretely, the International Civil Aviation Organization (ICAO).

Current realizations of Basic Access Control deploy symmetric cryptography (Triple-DES) and generate the corresponding encryption and authentication keys from passport information that is visible in the physical document (e.g., serial number, date of birth and expiration date). More concretly, the key derivation scheme (e.g., implemented in reader devices) includes three computations of SHA-1, one to derive the chip individual key K_Seed, and two consecutive computations that derive encryption key K_Enc and authentication key K_MAC. One of the main concerns pointed out by many experts is the low entropy of this visible information being insecure for key generation. The scheme has been already successfully attacked using offline dictionary attacks[6].

Using our hardware architecture this kind of attack can be mounted in much shorter time, and even real-time, i.e., the time needed to pass the inspection system. Note that the dictionary attack can be accelerated by pre-computing possible encryption keys using SHA-1 in advance. Then our hardware only has to check for a matching of ciphertexts implementing Triple-DES only.

Moreover, we are currently working on a device that can continuously read and record RF based communication at public places with high ePass density like airports. After the real-time decryption with our DES cracker, the information can be injected into distributed databases. Having installed such devices on many different airports and other similar places one can trace any person similar to tracing packages sent using postal services such as UPS.

3.3 Factorization

Since the introduction of public-key cryptography, the problem of factoring large composites is of increased interest. These days, the by far most popular asymmetric cryptosystem is RSA which was developed by Ronald Rivest, Adi Shamir and Leonard Adleman in 1977 [13]. The security of the RSA cryptosystem relies on the difficulty of factoring large numbers. Hence, the development of a fast factorization method could allow for cryptanalysis of RSA messages and signatures. The best known method for factoring large integers is the General Number-Field Sieve (GNFS). One important step within the GNFS is the factorization of mid-size numbers for smoothness testing, an efficient algorithm for which is the Elliptic Curve Method (ECM). Since ECM is suitable for parallelization, it is promising to be implemented in hardware.

The algorithm itself is almost ideal for improving the area-time product through special purpose hardware. First, it performs a very high number of operations on a very small set of input data, and is, thus, not very I/O intensive. Secondly, it requires relatively little memory. Thirdly, the operands needed for supporting GNFS are well beyond the width of current computer buses, arithmetic units, and registers, so that special purpose hardware can provide a much better fit. This justifies the higher development costs compared to a solution with DSPs. Lastly, it should be noted that the nature of the application allows for a very high degree of parallelization.

The first reported implementation of ECM in hardware was used to factor numbers of up to 200 bit [11]. However, the monetary cost of the used System-on-Chip hardware is quite high. Practical applications demand for a cheap realization of such ECM units. Therefore, a hardware platform consisting of many low-cost FPGAs seems to be an appropriate choice. As a result of the simple control logic for the ECM algorithm, no complex microcontroller is required and most logic can easily be put into the FPGA.

[6] Experiments on the Netherlands' epass demonstrated that the encrypted information can be revealed in 2 hours after intercepting the communication, see `http://www.riscure.com/news/passport.html`. The issuing scheme in the Netherlands has about 35 bits of entropy.

We propose to extend the proof-of-concept implementation of [11] to a highly parallel design comprised of many (cheap) FPGAs which can be used to assist attacks on RSA cryptosystems with moduli of sizes up to 1024 bit. For larger moduli, such a design demands for large quantities of ECM engines such that an ASIC implementation is preferable.

All algorithms are chosen such that they allow for an implementation with a low area consumption and a relatively high speed. At the time of writing, a basic ECM unit has been realized with a very efficient ALU and first performance results are available in Table 1 (the values include overhead for memory access). A single unit can be clocked with 40 MHz and requires

Table 1. ECM implementation (200 bit modulus) (Xilinx XC3S1000, 40MHz)

Operation	Time
modular addition/ subtraction	100 ns
modular multiplication/ squaring	5.13 μs
point addition	31.4 μs
point duplication	26.0 μs

approximately 40% of the slices of the Spartan3 device. Most memory has been realized with internal dual-port RAM.

3.4 Solving Elliptic Curve Discrete Logarithms

Besides factorization, many public-key cryptosystems are based on the difficulty of solving discrete logarithms in cyclic groups, known as the Discrete Logarithm Problem (DLP). A popular choice of such is the Elliptic Curve Cryptosystem (ECC) [7].

Attacking ECC requires the same algorithmic primitives as the cryptosystem itself, namely point addition and point doubling. Similar to the case of ECM in the previous section, these primitives can be implemented very efficiently in hardware. A parallel Pollard's Rho (PR) algorithm is described in [17].

The PR algorithm essentially does a great many computations without the necessity of communication. Only particular results have to be reported to a central control unit, which can be realized by, e.g., a host-PC connected to the FPGA. The parameterization of the algorithm can be optimized for a low area-time product and a low communication overhead. Hence, the reports of the PR units to the host occur not very frequently. As with the ECM unit, a single PR unit is comprised of an ALU, some memory and a control logic. The ALU for PR comprises modular inversion as additional function. Opposed to ECM, every single PR unit requires an individual control flow. Hence, the logic overhead for the algorithmic state machine is slightly higher. For curves defined over prime fields of 160 bit, two independent PR units can be loaded onto a XC3S1000 device. In this case, the maximum clock frequency is approximately 40 MHz and the area usage is 6067 slices (79%). With 160 bit curves, a point addition requires 846 cycles (21.15 μs) and 47 280 point operations can be performed per second by one unit. Consequently, a single FPGA can compute approximately 94 500 point operations per second.

We can parallelize Pollard's rho for COPACOBANA with the method presented in [17]. All instances of the algorithm can run independently from each other. Solely certain values have to be collected by a host-PC. Unlike ECM, we need a separate control logic for every single PR unit, yielding a slight overhead in logic on the FPGA. All units can be addressed individually by the host-PC using a unique address.

The chosen parameterization of the algorithm allows for a moderate communication overhead. In principle, all units compute point additions until they hit a point of a certain structure (so-called *distinguished points*). In such a case, the distinguished point is loaded to an output buffer for transmission to the host-PC while the computation continues.

For successfully solving the discrete logarithm problem over curves defined over prime fields $\mathbb{F}_p$, we have to compute approximately $\sqrt{q}$ points, where q is the largest prime power of the order of the curve. Appendix B provides estimates for solving the Certicom ECC challenges in hardware and software.

4 Conclusion and Future Work

The work at hand presents the design and first prototype of a cost-efficient hardware for running cryptanalytical algorithms. COPACOBANA can be built for for less than US$ 10,000 and hosts 120 low-cost FPGAs which can be adopted to any suitable task which is parallelizable and has low communication requirements. For instance, we demonstrated how the Data Encryption Standard (DES) can be broken within 9 days with the hardware at hand at an average rate of 48 billion keys per second.

We described how the DES implementation at hand can be used to attack the recently introduced machine readable travel documents. Furthermore, we introduced to two cryptanalytical algorithms which can be used to attack asymmetric algorithms. We propose a massively parallel implementation of the elliptic curve method for factorization. Building an efficient ECM machine is believed to speed-up the factorization of current RSA moduli. Furthermore, we can analyze the security of elliptic curve cryptosystems by solving the ECDLP with a hardware architecture for the parallel Pollard's rho algorithm.

Even though breaking full-size RSA (1024 bit or more) or elliptic curves (ECC with 160 bit or more) is out of reach with COPACOBANA, our machine provides for the first time a tool for a reliable security estimation of RSA and ECC. Even more relevant is the fact that resource constrained applications, in particular mobile devices, sometimes settle with shorter parameters, such as the 80 bit and 112 bit ECC systems recommended by the SECG standard, which become vulnerable with our machine. Also, assuming Moore's law, we can predict the security margin of RSA and ECC in the years to come.

Recapitulating, COPACOBANA is the first and currently the only available cost-efficient design to solve cryptanalytical challenges. COPACOBANA was intended to, but is not necessarily restricted to solving problems related to cryptanalysis. Almost certainly there will exist more interesting problems apart from cryptology, which can be solved efficiently with the design at hand. In an ongoing project, we plan to apply the Smith-Waterman algorithm [21, 12] for scanning sequences of DNA or RNA against databases.

Future work includes optimization of the parallel implementations of the presented cryptanalytical algorithms to guarantee the best possible throughput. Furthermore, it seems promising to mount a real-world attack on the ePass and on other cryptographically weak systems such as SHA-1 with help of COBACOBANA.

Acknowledgments: We like to thank the Xilinx Inc. for the generous donation of Spartan-3 FPGAs which formed the basis of our design. We are also indebted to Jean-Jacques Quisquater and François-Xavier Standaert of the Université Catholique de Louvain for making their high-speed DES design available. Furthermore, we would like to thank Kerstin Lemke and Ahmad-Reza Sadeghi for interesting discussions on the security of machine readable travel documents.

References

1. M. Blaze, W. Diffie, R. L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security: A Report by an Ad Hoc Group of Cryptographers and Computer Scientists. Technical report, January 1996. Available at `http://www.counterpane.com/keylength.html`.
2. Certicom Corporation. Certicom ECC Challenges, 2005. `http://www.certicom.com`.
3. CESYS GmbH. USB2FPGA Product Overview. `http://www.cesys.com`, January 2005.
4. R. Clayton and M. Bond. Experience Using a Low-Cost FPGA Design to Crack DES Keys. In B.S. Kaliski, C.K. Koc Cetin, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA*, volume 2523 of *series*, pages 579 – 592. Springer-Verlag, August 2002.
5. W. Diffie and M. E. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *COMPUTER*, 10(6):74–84, June 1977.
6. Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design.* O'Reilly & Associates Inc., July 1998.
7. D. R. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer Verlag, 2004.
8. A. Juels, D. Molnar, and D. Wagner. Security and privacy issues in e-passports. In *SecureComm 2005, First International Conference on Security and Privacy for Emerging Areas in Communication Networks, Athens, Greece*, September 2005.
9. G.S. Kc and P.A. Karger. Security and Privacy Issues in Machine Readable Travel Documents (MRTDs). RC 23575, IBM T. J. Watson Research Labs, April 2005.
10. NIST FIPS PUB 46-3. *Data Encryption Standard.* Federal Information Processing Standards, National Bureau of Standards, U.S. Department of Commerce, January 1977.
11. J. Pelzl, M. Šimka, T. Kleinjung, J. Franke, C. Priplata, C. Stahlke, M. Drutarovský, V. Fischer, and C. Paar. Area-Time Efficient Hardware Architecture for Factoring Integers with the Elliptic Curve Method. *IEE Proceedings Information Security*, 152(1):67–78, October 2005.
12. G. Pfeiffer, H. Kreft, and M. Schimmler. Hardware Enhanced Biosequence Alignment. In *International Conference on METMBS*, pages 11–17. CSREA Press, 2005.
13. 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.
14. G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat. Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-DES. In *Field-Programmable Logic and Applications - FPL*, pages 181–193, 2003.
15. RSA Laboratories. Announcements: The RSA Data Security Secret-Key Challenge. *CRYPTOBYTES*, 2(3):16, 1997. Available at `ftp://ftp.rsa.com/pub/cryptobytes/crypto2n3.pdf`.
16. University of California, Berkeley. Seti@Home Website, 2005. `http://setiathome.berkeley.edu/`.
17. P.C. van Oorschot and M.J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12(1):1–28, 1999.
18. M. J. Wiener. Efficient DES Key Search. In William R. Stallings, editor, *Practical Cryptography for Data Internetworks*, pages 31–79. IEEE Computer Society Press, 1996.
19. M. J. Wiener. Efficient DES Key Search: An Update. *CRYPTOBYTES*, 3(2):6–8, Autumn 1997.
20. Xilinx. Spartan-3 FPGA Family: Complete Data Sheet, DS099. `http://www.xilinx.com`, January 2005.
21. C.W. Yu, K.H. Kwong, K.H. Lee, and P.H.W. Leong. A Smith-Waterman Systolic Cell. In *Proceedings of the 13th International Workshop on Field Programmable Logic and Applications — FPL 2003*, pages 375–384. Springer, 2003.

A Realization of COPACOBANA

Figure 4 shows the realization of a single FPGA DIMM module as printed circuit. COPACOBANA with a 19 DIMM modules is depicted in Figure 5.

Fig. 4. FPGA module (4-layer printed circuit board)

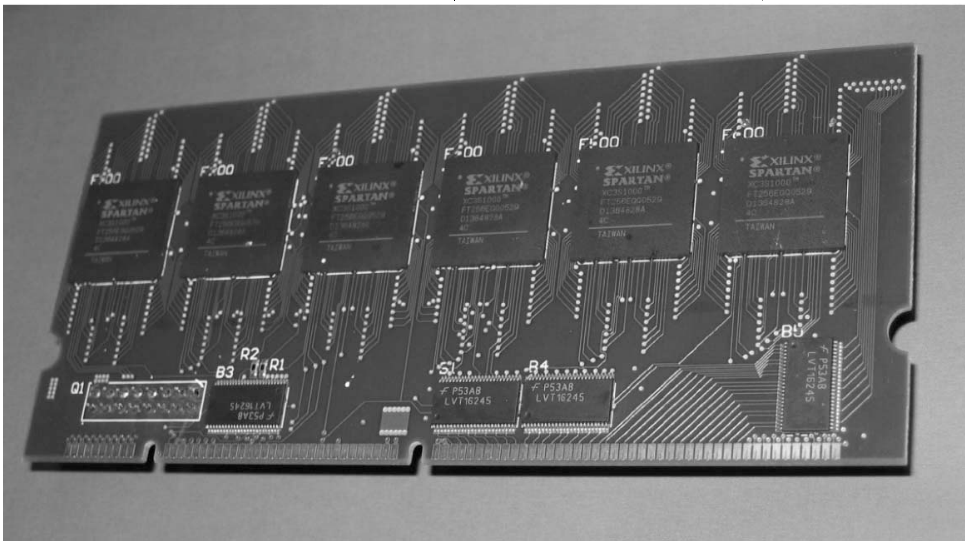
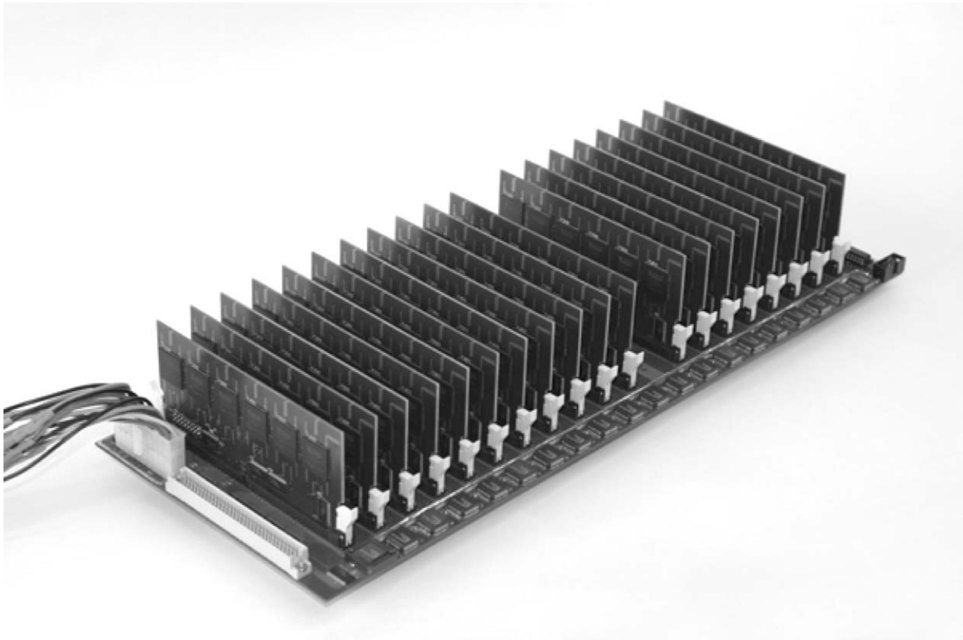


Fig. 5. COPACOBANA backplane with FPGA modules



B Certicom ECC Challenges

To show how secure ECC is (and, thus, how hard it is to solve the discrete logarithm problem on elliptic curves), the company certicom announced challenges for different bit-sizes [2]. The latest challenge solved was for a curve defined over a prime field of size of 109 bit. The challenge was estimated to take approximately $9 \cdot 10^6$ machine days on a conventional PC. For $q \approx 109$ bit, we would need to compute approximately 2^{54} point additions. In this case we would send every 2^{30}th point to a host-PC for subsequent comparisons. With COPACOBANA, the discrete logarithm could be solved in approximately 10^6 days with a single FPGA. Since the targeted FPGA is low-cost (approx. US$ 50 per piece at small quantities[7]), it is fair to assume that we can buy more than one FPGA for the price of a single PC. We assume that COPACOBANA can solve the challenge approximately 90 times faster than a PCs at equivalent costs.

Table 2 provides a comparison of the expected running time in days of a conventional PC versus the running time of COPACOBANA built of 120 FPGAs[8]. Furthermore, we assume the presence of 3, 2, and 1 PR unit(s) on a single FPGA for the bit-length 79-97, 109-191, and 239, respectively. Furthermore, a fixed clock rate of 40 MHz is assumed. The estimates for the machine days are taken from [2].

Table 2. Expected runtime on different platforms and for different Certicom ECC challenges

Challenge	**Pentium M@1.7GHz**	**COPACOBANA**
ECCp-79	49.0 d	0.13 d
ECCp-89	4.64 y	4.90 d
ECCp-97	74.7 y	93.4 d
ECCp-109	5570 y	24.2 y
ECCp-131	$1.40 \cdot 10^7$ y	$6.17 \cdot 10^4$ y
ECCp-163	$1.09 \cdot 10^{12}$ y	$7.63 \cdot 10^9$ y
ECCp-191	$2.17 \cdot 10^{16}$ y	$1.58 \cdot 10^{14}$ y
ECCp-239	$4.44 \cdot 10^{23}$ y	$7.18 \cdot 10^{21}$ y

[7] Xilinx offers this particular FPGA for US$ 12 at large quantities ($> 250,000$ pcs.).

[8] For simplicity, we neglect the central control instance and the required memory which is, in fact, the same for both the PC and the FPGA solution.