Three Years of Evolution: Cryptanalysis with COPACOBANA

Tim Güneysu[1], Gerd Pfeiffer[2], Christof Paar[1], Manfred Schimmler[2]
[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{gueneysu,cpaar}@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

In this paper, we review three years of development and improvements on COPACOBANA, the probably most popular, reconfigurable cluster system dedicated to the task of cryptanalysis. Latest changes on the architecture involve modifications for larger and more powerful FPGA devices with dedicated 32 MB of external RAM and point-to-point communication links for improved data throughput. We outline how advanced cryptanalytic applications, such as Time-Memory Tradeoff (TMTO) attacks or attacks on asymmetric cryptosystems, can benefit from these new architectural improvements.

1 Introduction

The security of symmetric and asymmetric ciphers is usually determined by the size of their security parameters, in particular the key-length. Hence, when designing a cryptosystem, these parameters need to be chosen according to the assumed computational capabilities of an attacker. Depending on the chosen security margin, many cryptosystems are potentially vulnerable to attacks when the attacker's computational power increases unexpectedly. In real life, the limiting factor of an attacker is often the financial resources. Thus, it is quite crucial from a cryptographic point of view to not only investigate the complexity of an attack, but also to study possibilities to lower the cost-performance ratio of attack hardware. For instance, a cost-performance improvement of an attack machine by a factor of 1000 effectively reduces the key lengths of a symmetric cipher by roughly 10 bit (since $1000 \approx 2^{10}$). Many cryptanalytical schemes spend their computations in independent operations, which allows for a high degree of parallelism. Such parallel functionality can be realized by individual hardware blocks that operate simultaneously, improving the running time of the overall computation by a perfect linear factor. At this point, it should be remarked that the high non-recurring engineering costs for ASICs have put most projects for building special-purpose hardware for cryptanalysis out of reach for commercial or research institutions. However, with the recent advent of low-cost programmable ICs which host vast amounts of logic resources, special-purpose cryptanalytical machines have now become a possibility outside government agencies.

In this work we review the evolution of a special-purpose hardware system which provides a cost-performance that can be significantly better than that of recent PCs (e.g., for the exhaustive key search on DES). The hardware architecture of this Cost-Optimized Parallel Code Breaker (COPACOBANA) was initially introduced on the SHARCS and CHES workshops in 2006 [24, 25]. In this contribution we will describe further research on cryptanalytical applications over

the last three years and ongoing improvements on the hardware to cope with new requirements of these advanced applications.

The original prototype of the COPACOBANA cluster consists of up to 120 FPGA nodes which are connected by a shared bus providing an aggregate bandwidth of 1.6 Gbps on the backplane of the machine. COPACOBANA is not equipped with dedicated memory modules, but offers a limited number of RAM blocks inside each FPGA. Furthermore, COPACOBANA is connected to a host PC with a single interface to control all operations and provide a little amount of I/O data.

In the following sections, we present cryptanalytic case studies for a large variety of attacks which all make use of the COPACOBANA cluster system. Examples for these case studies include exhaustive key search attacks on the Data Encryption Standard (DES) blockcipher and related systems (e.g., Norton Diskreet), the electronic passport as well as the GSM mobile phone encryption based on the A5/1 streamcipher. More advanced attacks with COPACOBANA comprise implementations for integer factorization (with the Elliptic Curve Method), computations on elliptic curve discrete logarithms and Time-Memory Tradeoffs (TMTO). We briefly review attack implementations and compile a list of improvements on hardware level that can lead to improved performance. Finally, we present a modified cluster architecture which addresses most of the determined issues and promises excellent performance results for the next generation of cryptanalytic applications.

The paper is structured as follows: we begin with a brief review of the original COPACOBANA architecture and a list of case studies on cryptanalytic attacks in Section 3. Next, we present the modified cluster architecture with improvements based on our findings in the previous section. We conclude with an outlook on future cryptanalysis based on COPACOBANA.

2 Architecture of COPACOBANA

Our first prototype of an Cost-Optimized Parallel Code Breaker (COPACOBANA) was produced for less than € 10,000 (material and manufacturing costs only). It was primarily designed for applications and simple cryptanalytic attacks with high compuational complexity but minimal requirements on communications and local memory. In addition to that, it assumes that the computationally expensive operations are inherently parallelizable, i.e., single parallel instances do not need to communicate with each other. The design for limited communication bandwidth was driven by the fact that the computation phase heavily outweighs the data input and output phases. In fact, COPACOBANA was designed for applications in which processes are computing most of the time, without any input or output. Communication was assumed to be 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 FPGAs on the cluster by a simple interface. Furthermore, simple brute-force attacks typically demand for very little memory so that we considered the available memory on low-cost FPGAs (such as the Xilinx Spartan-3 devices) to be sufficient.

Recapitulating, COPACOBANA consists of many independent low-cost FPGAs, connected to a host-PC via a standard interface, e.g., USB or Ethernet. The benefit of such a standard interface is the easy scalability and to attach more than one COPACOBANA device to a single host-PC. Note that the initialization, control of FPGAs, and the accumulation of results is done by the host. All time-critical computations such as the cryptanalytical core tasks are performed

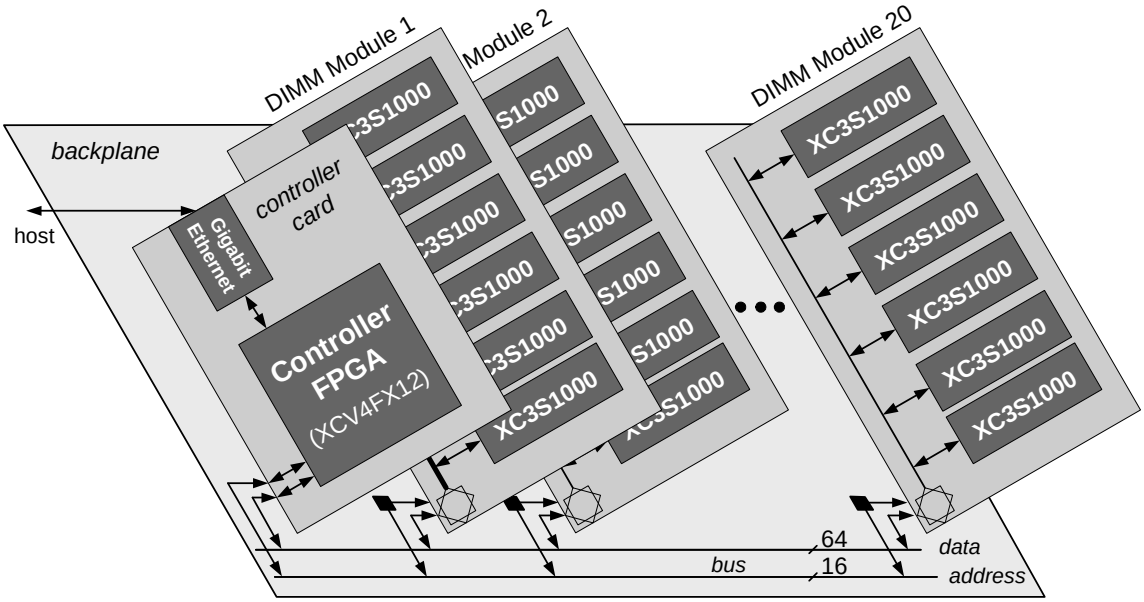

Figure 1: Architecture of COPACOBANA

on the FPGAs. The first prototype of COPACOBANA was equipped with up to 120 FPGAs, distributed along 20 slots - in FPGA modules which can be plugged into a single backplane. Note that the choice for 120 FPGAs was driven by the form factor of the FPGA module which was designed according to cheap and standardized DIMM interface specifications. One (single-sided) DIMM-sized FPGA module can host 6 FPGA devices in a 17×17 mm package, such as the Xilinx Spartan3-1000 FPGA (XC3S1000, speed grade -4, FT256 packaging). This device 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) [40]. The backplane of COPACOBANA connects all FPGA devices with a shared 64-bit data and 16-bit address bus. The entire cluster system is depicted in Figure 1.

COPACOBANA was designed for single master bus arbitration for simplified control. However, in case the communication scheduling of an application is not predictable, the bus master is required to poll all FPGAs for new events and returned data. This significantly slows down the communication performance and increases latencies when reading data back from the FPGAs. Data transfer from and to the FPGAs is accomplished by a dedicated control unit. Originally, we decided to pick a small development board with an FPGA (CESYS USB2FPGA [6]) in favor of a flexible design. The board provides an easy-pluggable 96-pin connector which we use for the connection to the backplane. In later versions of COPACOBANA, we replaced the USB controller using an TCP/IP-based unit so that COPACOBANA can be controlled remotely and can be placed externally, for example in a server room.

3 Cryptanalytic Applications for COPACOBANA

In this section, we briefly describe cryptanalytic applications which we have already implemented on our initial release of the COPACOBANA cluster system. We compiled the most important

facts and key points of each application into individual case studies which eventually should help to identify shortcomings and potential enhancements of our cluster architecture.

3.1 Exhaustive Key Search Scenarios

In the following sections, we present a short survey about our work on exhaustive key search and guessing attacks on a variety of real-world systems. Since all these applications consist mostly out of very basic tasks that can be efficiently parallelized on completely independent computational cores, they can be perfectly mapped onto a highly parallel cluster system such as COPACOBANA.

3.1.1 Case-Study I: Breaking DES with Exhaustive Search

Our first cryptanalytic target application was the exhaustive key search on the DES block cipher. We implemented a known-plaintext attack and used an improved version of the DES engine of the Université Catholique de Louvain's Crypto Group [36] as a core component. Inside a single FPGA, we could place four of such DES engines which allows for sharing plaintext-ciphertext input pairs and the key space. Our first implementation and successful attack was presented in [25]. Since this original publication, we were able to improve the system performance by use of additional pipelined comparators and simplified control logic. Now, we are able to operate each of the FPGAs at an increased clock rate of 136 MHz with a overall gain in performance by 36%, compared to [25]. Consequently, 2^{42} keys can be checked in $2^{40} \times 7.35\,\text{ns}$ by a single FPGA, which is approximately 135 minutes. Since COPACOBANA hosts 120 of these low-cost FPGAs, the entire system can check $4 \times 120 = 480$ keys every 7.35 ns, i.e., 65.28 billion keys per second. To find the correct key, COPACOBANA has to search through an average of 2^{55} different keys. Thus, it can find the right key in approximately $T = 6.4$ days on average. By increasing the number n of COPACOBANAs used for this task, we can further decrease this average runtime of the attack by a linear factor $1/n$.

3.1.2 Case-Study II: Cracking Norton Diskreet

In the 1990s, Norton Diskreet, a part of the well-known Norton Utilities package, was a very popular encryption tool. Diskreet can be used to encrypt single files as well as to create and manage encrypted virtual disks. The tool provides two encryption algorithms one can choose from, a (cryptographically weak) proprietary algorithm and the DES in cipher block chaining (CBC) mode. To encrypt a file or virtual disk, Diskreet asks for a password with a minimal length of 6 and a maximal length of 40 bytes. From this password the 56-bit DES-key is generated. The password-to-key mapping works as follows: First, leading whitespace characters are removed before the password is converted to uppercase characters which are divided into chunks of 8 bytes. Then, all 8-byte blocks are subsequently XORed with each other and the resulting sum is used as DES-key. Obviously, this method of key generation is unfavorable since the password-to-key mapping is not chaotic at all. As explained more thoroughly in [13], we can modify the key generator to our implementation (cf. Section 3.1.1) so that it generates only a limited set of keys according the password distribution based on different assumptions. The performance of the attack with a single COPACOBANA is shown in Table 1.

Key space	Remark	DES decryptions (on average)	Runtime
$\{A, \ldots, Z, @, [, \backslash,], \hat{}, \_\}$	Known pwd length	2^{31}	32.8 μs
$\{A, \ldots, Z, @, [, \backslash,], \hat{}, \_\}$	Unknown pwd length	2^{35}	0.53 s
7-bit ASCII	8 Characters	2^{47}	35.9 m
8-bit ASCII	8 Characters	2^{55}	6.39 d

Table 1: Breaking Norton Diskreet with COPACOBANA

3.1.3 Case-Study III: Hacking the ePassport

The electronic passport (ePass), as specified by the international civil aviation organization (ICAO), is deployed in many countries all over the world. The security and privacy threats have been widely discussed (e.g., [20], [22], [17], [19]). A chip embedded in the machine readable travel document (MRTD) contains private data as text, such as name, date of birth and sex, as well as biometrics [29]. A digital facial photograph and in some countries additionally fingerprints or an iris scan of the passport holder can be accessed via a contactless interface based on the ISO 14443 [18] standard. The wireless communication constitutes new opportunities for attackers, such as relay attacks [21] or eavesdropping from a range of several meters, as investigated in [14], [35] and [9]. To prevent unauthorized access to the information transferred via the radio frequency (RF) interface some countries, among them Germany and the Netherlands, employ the so-called basic access control (BAC). The BAC is meant to secure the interchanged data, i.e., establish a confidential channel, by employing symmetric cryptography. The secret keys needed for carrying out the BAC are stored in the embedded integrated circuit (IC) and can also be derived from a machine readable zone (MRZ) that is printed on the paper document.

We here shortly sketch a possible attack on the BAC using COPACOBANA which is adapted from [4] and more thoroughly described in [27, 13]. We assume that a device for eavesdropping of the RF field can be mounted nearby an e-passport inspection system, such that all bits transmitted via the air channel can be captured and stored in a database. The fundamental secret required to access the private data rely on an authentication key k_{MAC} and an encryption key k_{ENC} that are derived from the MRZ information on the paper document according to

$$k_i = msb_{16}\left(\text{SHA-1}\left(msb_{16}\left(\text{SHA-1}\left(\text{MRZ}\right)\right) \| C\right)\right),$$

where the $msb_{16}(x)$ function selects the most significant 16 byte of x. After the first execution of SHA-1[32], the result is concatenated with a constant C which is either $C =$ `0x00000001` for k_{ENC} or $C =$ `0x00000002` for k_{MAC}. The keys are then used with a Triple DES (TDES) block-cipher [30]. Having access to system messages by eavesdropping the near-field communications, we can attack the combination of SHA-1 and TDES in a brute-force attack scenario (although the theoretical keyspace available to TDES is out of reach for conventional exhaustive key search attacks). This is possible since the entropy of the MRZ can be found to be as low as $\approx 2^{33}$ for realistic scenarios based on the BAC realizations of the Netherlands and Germany [35, 27]. Hence, instead of performing an exhaustive search on every possible TDES key, we implemented again a smart generator which only produces a limited set of outputs that are reasonable MRZ values.

The time critical component in our attack implementation is the SHA-1 hash function, determining the maximum clock frequency of $f_{clk} = 40\,\text{MHz}$ and requiring 80 clock cycles for

one key candidate. Processing of one key thus requires $80 \times 25\,\text{ns} = 2\,\mu\text{s}$. As there are 120 FPGAs running in parallel, each consisting of four encryption engines, $4 \times 120 = 480$ keys are tested every $2\,\mu$s, resulting in a throughput of $2^{27.84} \approx 240$ million keys per second. On average, testing of 2^{33} keys reveals the correct candidate in $\frac{2^{32}}{2^{27.84}} \approx 18$ seconds which can be regarded as real-time, compared to the duration of one inspection at the border control.

3.1.4 Case-Study IV: Breaking the A5/1 Streamcipher

A5/1 is a synchronous stream cipher that is used for protecting GSM communication. In the GSM protocol, the communication channel is organized in 114-bit frames that are encrypted by XORing them with 114-bit blocks of the keystream produced by the cipher as follows: A5/1 is based on three LFSRs, that are irregularly clocked. The three registers are 23, 22 and 19 bits long, representing the internal 64-bit state of the cipher. During initialization, a 64-bit key k is clocked in, followed by a 22-bit initialization vector that is derived from the publicly known frame number. After that a warm-up phase is performed where the cipher is clocked 100 times and the output is discarded. For a detailed description of A5/1 please refer to [3].

Most of previously proposed attacks against A5/1 lack from practicability and/or have never been fully implemented. In contrast to these attacks, we present in [11] a real-world attack revealing the internal state of A5/1 in about 6 hours on average (and about 12 hours in the worst-case) using COPACOBANA. The implementation is an optimization of a *guess-and-determine* attack as proposed in [23], including an improvement in runtime of about 13% compared to their original approach. Each FPGA contains 23 guessing engines running in parallel at a clock frequencey of 104 MHz each. To mount the attack, only 64 consecutive bits of a known keystream are required and we do not need any precomputed data. Note, however, that an average of 6 hours runtime still cannot be considered a real-time attack when using a single COPACOBANA. In this case, we need to record the full communication first and attack its encryption offline afterwards. Alternatively, by adding further machines the attack time will be linearly reduced, e.g., 100 machines only require 3.6 minutes for a sucessfull attack on average.

3.2 Advanced Cryptanalytic Applications

In the last section, we briefly surveyed simple attacks based on exhaustive key searches or guessing. All these attacks have in common that their performance is basically limited by the number of computations. In other words, the available logic of the FPGA devices on COPACOBANA directly determines the performance of the attack. By incrementing the number of COPACOBANA units we yield a speed-up in performance by a perfect linear factor. This, however, does not hold for the following, more advanced attacks.

3.2.1 Case-Study V: Time-Memory (Data) Tradeoffs

Time-Memory Tradeoff (TMTO) and Time-Memory-Data Tradeoff (TMDTO) methods were designed as a compromise between the two well-known extreme approaches: either to perform an exhaustive search on the entire key space of the cipher or precomputing exhaustive tables representing all possible combinations of keys and ciphertexts for a given plaintext. The TMTO and TMDTO strategies offer a way to reasonably reduce the actual search complexity (by doing some kind of precomputation) while keeping the amount of precomputed data reasonably low, whereas "reasonably" has to be defined more precisely. Roughly speaking, it depends on the concrete attack scenario (e.g., real-time attack), the internal step function and the available resources for the precomputation and online (search) phase.

Method	DU [GB]	PT (COPA) [days]	OT [ops]	TA	SR
Hellman	1897	24	$2^{40.2}$	$2^{40.2}$	0.80
Rivest	1690	95	2^{21}	$2^{39.7}$	0.80
Oechslin	1820	23	$2^{21.8}$	$2^{40.3}$	0.80

Table 2: TMTO methods according to: expected runtimes and memory requirements using COPACOBANA for precomputations.

Existing TMTO methods by Hellman, Rivest and Oechslin [16, 7, 31] share the natural property that in order to achieve a significant success rate much precomputation effort is required on chained computations. A representation of start point and end point of each chain is stored in (a set of) large tables, e.g., on hard disk drives. The actual attack takes place in a second search phase (online phase) in which another chain computation is performed on the actual data and compared to the stored endpoints in the tables. In case a matching endpoint is found in the table, the sequence of keys can be reconstructed using the corresponding start point. There are few contributions attacking DES with the TMTO approach. In [38] an FPGA design for an attack on a 40-bit DES variant using Rivest's TMTO method [7] was proposed. In [28] a hardware architecture for UNIX password cracking based on Oechslin's method [31] was presented. However, to the best of our knowledge, a set of complete TMTO precomputation tables for *full* 56-bit DES was never created up to now.

The idea of cryptanalytic TMDTO is due to Babbage, Biryukov and Shamir [1, 2]. TMDTOs are variants of TMTOs exploiting a scenario where multiple data points $y_1, \ldots, y_D$ of the function g are given and one has just to be successful in finding a preimage of one of them. Such a scenario typically arises in the cryptanalysis of stream ciphers where we like to invert the function mapping the internal state (consisting of $\log_2(N)$ bits) to the first $\log_2(N)$ output bits of the cipher produced from this state. We employed this method to mount an advanced attack on the A5/1 streamcipher which provides a runtime of far less than 6 hours (cf. Section 3.1.4), at the cost of a reduced success probability.

In [13] we present possible configurations and parameters to use COPACOBANA for TMTO/TMDTO precomputations both to attack the DES blockcipher and the A5/1 streamcipher. Our estimates took the assumed communication bandwidth between host-PC and backplane of 24 MBit/s into account. To break DES with TMTOs on COPACOBANA, Table 2 presents our worst case expectations concerning success rate (SR), disk usage (DU), the duration of the precomputation phase (PT) as well as the number of table accesses (TA) and calculations (C) during the online phase (OT). Note that for this extrapolation, we have used again the implementation of our exhaustive key search on DES (cf. Section 3.1.1).

For the implementation of the TMDTO attack on A5/1, we selected the set of parameters presented in the second row of Table 3, since it produces a reasonable precomputation time and a reasonable size of the tables as well as a relatively small number of table accesses. The success rate of 63% may seem to be small, but it increases significantly if more data samples are available: For instance, if 4 frames of known keystream are available, then $D = 4 \cdot 51 = 204$ and thus the success rate is increased to 96%.

Although both attacks are realistic (precomputations of less than one and about 3 months for DES and A5/1 respectively), we encountered several issues while running the attacks. One

m	S	d	I_ℓ	PT [days]	DU [TB]	OT [secs]	TA	SR [%]
2^{41}	2^{15}	5	$[2^3,2^6]$	337.5	7.49	27.8	2^{21}	0.86
2^{40}	2^{14}	5	$[2^4,2^7]$	95.4	4.85	10.9	2^{20}	0.63
2^{39}	2^{15}	5	$[2^3,2^6]$	84.4	3.48	27.8	2^{21}	0.60
2^{37}	2^{15}	6	$[2^4,2^8]$	47.7	0.79	73.5	2^{21}	0.42

Table 3: A5/1 TMDTO: Expected runtimes and memory requirements. The choice and explanation for parameters are described thoroughly in [13]

problem is the access time to hard disk storage in the online phase which is not reflected in the tables above and takes a considerable amount of time for itself (about 4-10 ms per access). Moreover, the generation of precomputation tables is slower than expected with respect to the communication link between host-PC and COPACOBANA backplane. It turned out that the communication speed of 24 MBit/s must be considered as theoretical throughput limit since additional data overhead and latencies need to be taken into account as well. In other words, to support TMTO/TMDTO for required parameters, the communication facilities of COPACOBANA need to be improved by at least one order of magnitude.

3.2.2 Case-Study VI: Integer Factorization

The factorization of a large composite integer n is a well-known mathematical problem which has attracted special attention since the invention of public key cryptography. RSA is known as the most popular asymmetric cryptosystem and was originally developed by Ronald Rivest, Adi Shamir and Leonard Adleman in 1977 [34]. Since the security of RSA relies on the attacker's inability to factor large numbers, the development of a fast factorization method could allow for cryptanalysis of RSA messages and signatures. Recently, the best known method for factoring large RSA integers is the General Number-Field Sieve (GNFS). An important step in the GNFS algorithm is the factorization of mid-sized numbers for smoothness testing. For this purpose, the Elliptic Curve Method (ECM) has been proposed by Lenstra [26] which has been proved to be suitable for parallel hardware architectures in [37, 10, 8], particularly on FPGAs.

The ECM algorithm performs a very high number of operations on a very small set of input data, and is not demanding in terms of high communication bandwidth. Furthermore, the implementation of the first stage of ECM requires only little memory since it is based on point multiplication on an elliptic curve. The operands required 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 solution.

In [8] it has been shown that the utilization of DSP slices in Virtex-4 FPGAs for implementing a Montgomery multiplication can significantly improve the ECM performance. In that work, the authors used a fully parallel multiplier implementation which provides the best known performance figures for ECM phase 1 so far, however they did provide details how to realize ECM phase 2.

To accelerate integer arithmetic using a similar strategy, we designed a new slot-in module for use with a second release of COPACOBANA hosting 8 Xilinx Virtex-4 XC4VSX35 FPGAs, each providing 192 DSP slices. Due to the larger physical package of the FPGAs (FF668 package with dimension of 27x27 mm) we enlarged the modules. This included also modifications of the

Aspect	Our work	Results [10]
Modular Adder/Subtracter	14 clk	31 clk[1]
Montgomery Multiplication	118 clk	167 clk[1]
Point Doubling	434 clk	n/a
Point Addition	500 clk	n/a
Combined Point Doubling and Addition	689 clk	947 clk[1]
Clock Frequency of an ECM Core	200 MHz	135 MHz

Table 4: Clock cycles and frequency for point multiplication of 151-bit numbers required in phase 1 of ECM on Virtex-4 devices (single core)

corresponding connectors on the backplane. For more efficient heat dissipation at high clock frequencies up to 400 MHz, an actively ventilated heat sink is attached to each FPGA. With a more powerful power supply providing 1.5 kW at 12 V, we could run a total of 128 Virtex 4 SX35 FPGAs distributed over 16 plug-in modules.

In contrast to [8], we used a multi-core ECM design per FPGA. A single ECM engine comprises of an arithmetic unit computing modular multiplication and additions, a point multiplication unit for phase 1 and ROM tables for phase 2. Only point operations on the elliptic curve are performed on FPGAs, this means that the setup of the Montgomery curve needs to be done on the host-PC and then transferred to the FPGAs. We provide first estimates for ECM phase 1 shown in Table 4 and compare our results to the implementation presented in [10].

Although the implementation based on DSP slices promise better results on Virtex-4 FPGAs compared to [10], we like to point out that the switch to Virtex-4 SX35 devices has a strong negative effect on our cost-performance ratio. With respect to a Spartan-3 device, a single Virtex-4 SX35 is much more expensive (roughly a factor of 10-20) and does not outperform the cheaper Spartan-3 devices by an corresponding factor. Hence, we still consider Spartan-3 devices more appropriate for cryptanalytical tasks due to their better cost-performance ratio. In particular, latest Spartan-3A DSP and Spartan-6 devices also come with a number of DSP slices, so that expensive Virtex devices (which formerly were the only devices with DSP slices) are not a necessity anymore.

Another issue of ECM is memory. Although ECM stage 1 has only very moderate memory constraints it can be considerately improved by additional computations in a second stage. However, stage 2 involves a significant amount of precomputations as well as storage for prime numbers. Since memory on FPGA devices is rather limited (192×18 KBit BRAM elements per device), a fast accessible, external memory could help to improve the beneficial effect of stage 2 by storing even larger tables.

3.2.3 Case-Study VII: Solving Elliptic Curve Discrete Logarithms

Another popular problem used for building public-key cryptosystems is known as the Discrete Logarithm Problem (DLP) where the exponent ℓ should be determined for a given $a^\ell \mod n$. A popular derivative is the Elliptic Curve Discrete Logarithm Problem (ECDLP) for Elliptic Curve Cryptosystems (ECC) [15].

[1]The presented cycle count for 151-bit integers was scaled down accordingly to the results given in [10] for 198-bit parameters. Here, Gaj *et al.* reported their implementation to take 1212 cycles for one combined 198 bit point doubling and addition (ECM stage 1).

An attack on ECC relies on the same algorithmic primitives as the crypto system itself, namely point addition and point doubling. Up to now, the best known algorithm for this purpose is the Pollard's Rho (PR) algorithm for parallel implementation described in [39]. This variant of the original PR method [33] allows for a linear gain in performance with the number of available processors. This can be efficiently implemented in hardware as presented in [12].

The PR algorithm essentially determines distinguished points on the elliptic curve. These points are reported to a central host computer which awaits a collision of two points. A distinguished point is defined to be a point with a specific characteristic, e.g., its x-coordinate has a fixed number of leading zero bits. To reach such a distinguished point, PR follows a so called pseudo-random walk on the elliptic curve by subsequently adding points from a fixed, finite set of random points. Hence, with careful parametrization of the distinguished point criterion, the duration of a computation until a distinguished point is found can be adapted to the bandwidth constraints of the system. Furthermore, the PR does not need a large memory for computation so that the COPACOBANA system seems to be a suitable platform for running the algorithm. As with the ECM unit, a single PR unit is comprised of an arithmetic unit, a few kilobytes of RAM and control logic. The arithmetic unit supports modular inversion as an additional function required for uniquely determining distinguished points.

For a parallelized PR on COPACOBANA according to the method presented in [39], all instances of the algorithm can run completely independent from each other. For 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. Note that the transfer of data between host computer and point processing units on the FPGA can be performed independently from the computations.

Implementing the PR on Spartan-3 FPGAs for solving the ECDLP over curves with a length of 160 bits and using an affine point representation, we achieve a maximum clock frequency of approximately 40 MHz and an area usage of 6067 slices (79%) for two parallel instances. The corresponding point addition requires 846 cycles so that slightly less than 50,000 point operations can be performed per second by one unit. Consequently, a single COPACOBANA can compute about 11.3 million point operations per second. Table 5 compares our results for COPACOBANA with challenges and corresponding estimates from Certicom based on the computing time of an (outdated) Intel Pentium 100. To compare our results with COPACOBANA against more recent systems, we refer to the solved ECC P-109 challenge that took 10,000 computers (mostly PCs) running 24 hours a day for a total time of 549 days. To solve this challenge in the same time, according to Table 5 about 17 COPACOBANAs (and subsequently an investment of € 170,000) would be required. In return, assuming a single PC of the original cluster to cost only about € 200, this already sums up to the amount of € 2 million, excluding any additional operational costs for power and cooling.
Note that our Pollard-Rho implementation also can benefit from advanced FPGAs with DSP slices (cf. Section 3.2.2. The large n-bit adders can be more efficiently implemented using cascades of the 48-bit adder contained in each DSP slice.

4 Enhancing the COPACOBANA Architecture

Recapitulating the issues of our COPACOBANA architecture according to the application shown in Section 3, we should consider a redesign to meet the following requirements:

- *Larger FPGA Devices:* more logical elements on an FPGA enable more complex applications or more computational cores per device.

k	**Certicom Est.** [5]	**Single XC3S1000**	**Single COPACOBANA**
79	146 d	15.3 d	3.06 h
89	12.0 y	1.62 y	4.93 d
97	197 y	30.7 y	93.4 d
109	$2.47 \cdot 10^4$ y	$2.91 \cdot 10^3$ y	24.3 y
131	$6.30 \cdot 10^7$ y	$7.40 \cdot 10^6$ y	$6.17 \cdot 10^4$ y
163	$6.30 \cdot 10^{12}$ y	$9.15 \cdot 10^{11}$ y	$7.62 \cdot 10^9$ y
191	$1.32 \cdot 10^{17}$ y	$1.89 \cdot 10^{16}$ y	$1.57 \cdot 10^{14}$ y
239	$3.84 \cdot 10^{24}$ y	$8.62 \cdot 10^{23}$ y	$7.18 \cdot 10^{21}$ y

Table 5: Expected runtime on different platforms and for different Certicom ECC challenges

- *Local Memory per FPGA:* a few megabytes of fast SRAM should be placed adjacent to each FPGA device to provide additional storage while solving more complex problems.
- *Dedicated Communication Links*: Only a single access to an FPGA at a time was possible in the recent communication models. Individual links for each FPGA simplify data communication and also enable high performance from simultaneous data exchange.
- *Improved Controller between Host-PC and COPACOBANA*: the main bottleneck with respect to data communication is the controller interface between backplane and host-PC. The embedded controllers for USB and Gigabit Ethernet, which we used for previous controller interfaces, did not provide sufficient performance due to high overhead (e.g., TCP/IP and USB frame packaging). A solution could be to integrate the host-PC inside the COPACOBANA case and directly link the PC's mainboard with the backplane, e.g., using a high-speed PCIe link.
- *Improved Signaling*: the single-ended I/O lines for the data and address bus system were subject of significant noise and side-effects like signal crosstalk. Improved signal quality can be achieved by switching to differential I/O at the cost of another signal line per I/O port. This is also beneficial when attempting to increase the data transmission frequency.

To address the aspects mentioned above, we entirely redesigned the backplane and FPGA module of our cluster system. The new FPGA module consists of 8 FPGAs with a package size up to $27 \times 27mm$, a CPLD for system control (e.g., temperature and voltage monitoring) and DC/DC converters (dependant on the FPGA type used). Most suitable FPGA devices for the new systems are the low-cost Xilinx Spartan-3 5000 with up to 74,880 logic cells (104 hardware multipliers, 104 BRAMs, FG676 packaging), the Xilinx Spartan-3A DSP 3400 with 53,712 logic cells (126 DSP48A, 126 BRAMs, FG676 packaging) and the upcoming class of Spartan-6 FPGAs. For the first prototype of the enhanced COPACOBANA, we chose 128 Spartan-3 5000 devices. Moreover, we placed 32 MB of SRAM adjacent to each FPGA that can be accessed by the FGPA within a single clock cycle at 100 MHz.

The new COPACOBANA design integrates the host-PC (uATX format) in the same case so that short (and fast) interfaces can be used. Target applications like for example Time-Memory Tradeoffs (cf. Section 3.2.1) require a high data rate between COPACOBANA and an external hard disk drive. One option could be to place a hard disk drive physically inside COPACOBANA, e.g., attached via a SATA interface to the integrated host-PC. The second option would let the integrated host-PC access other IT-infrastructure like a Storage-Area Network by additional interfaces or its two Gigabit Ethernet ports.

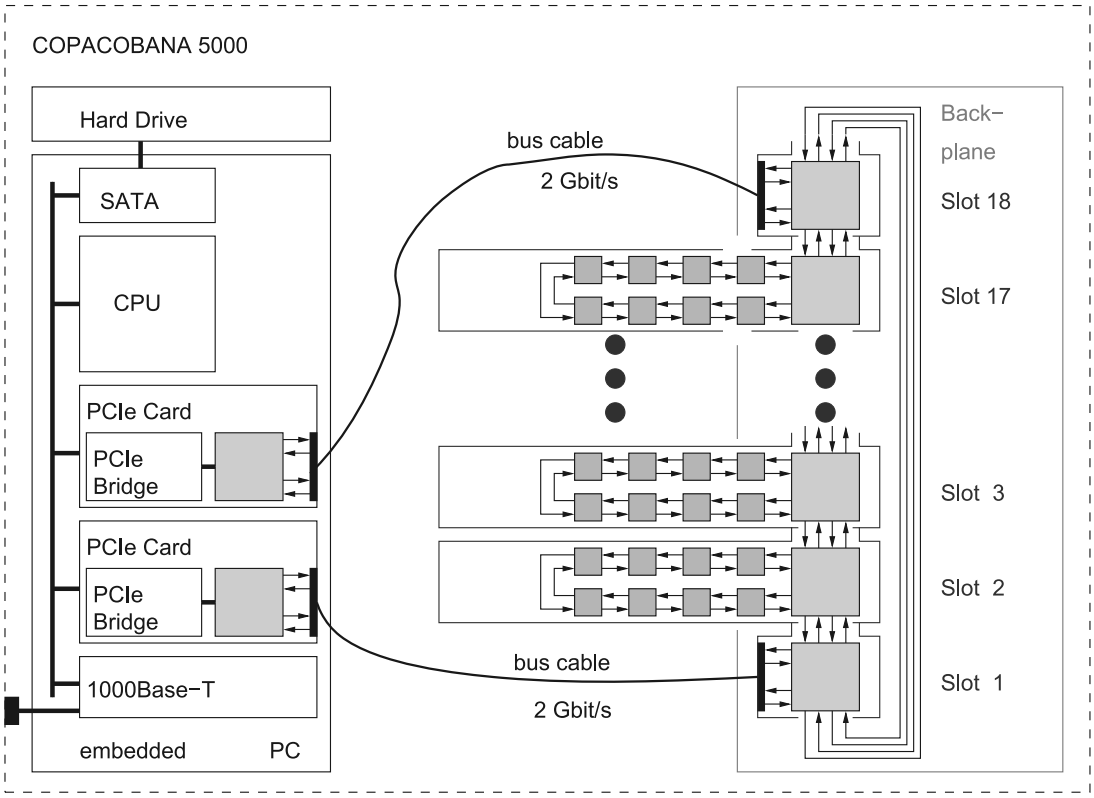

Figure 2: Enhanced COPACOBANA Architecture based on Xilinx Spartan-3 5000 FPGAs

The integrated PC connects to the FPGA-cluster by one or two PCIe modules. Therefore, the former communication bottleneck due to the single controller is completely eliminated. The new enhanced architecture COPACOBANA consists of an 18 slot backplane equipped with 16 FPGA-cards and two controller cards which connect the FPGA backplane to the integrated PC. Additional components of the new system are the $1.5kW$ main power supply unit (with 125A at 12V), six high-performance fans and a 19-inch rack of three hight units for the housing.

There are fast serial point-to-point connections between every two neighbors building a chain of FPGAs. In the configuration as described below, we employed eight Xilinx Spartan-3 5000 which are arranged as a systolic one-dimensional array, i.e., in each clock cycle data is transferred from one FPGA to the next in pipeline fashion according to a global, synchronous clock. Note that transferring data using a systolic array introduces significant latencies on the data path. However, since the target applications do not have real-time requirements, this should not be an issue for our cryptanalytic applications where operations usally can be interleaved to hide any latencies. Between the controller-cards and the integrated PC the maximum data rate is limited to 250 MByte/s due to the limitations of the PCIe connection. For high throughput, the I/O capabilities of the FPGA has to be considered carefully. The highest throughput can be achieved by connecting communicating chips by short point-to-point lines. To allow efficient broadcasts to all FPGAs simultaneously, there is a direct connection between adjacent FPGA-cards. The point-to-point interconnections consist of 8 pairs of wires in each direction. Each pair is driven by low voltage differential signaling (LVDS) with a speed of 250MHz, thus achieving a data-rate of 2Gbit/s. Figure 2 shows the overall architecture of our enhanced COPACOBANA.

As in the original system, the backplane connects to all FPGA cards on which individually the clock signals, data and power are (re-)generated and distributed. However, the bus is

managed now cooperatively by all FPGA modules instead of a single bus master, i.e., a central controller. Each FPGA module can transfer incoming data to the next slot or it can remove the data out of the stream. In this case, an empty data frame cycles through the bus pipeline from slot to slot. Note that another card is allowed to insert new data now into this empty slot. The two counter-rotating systolic datapaths allow to minimize the worst case latency to half of the total number of slots multiplied by the clock cycle time. The enhanced bus system assigns one ascending and one descending slot to each single card. This leads to a ring of point to point connections in which the bus system can be seen as a circular, parallel shift register.

Due to the modular architecture further developments can be incorporated seamlessly. For example, alternative FPGA modules equipped with a Spartan-3A DSP 3400 or Spartan-6 FPGAs can easily plugged into the backplane. In particular, it is possible to run even a heterogenous configuration, with a mixed set of FPGAs for different tasks.

5 Expected Improvements in Cryptanalytical Applications

At the time of writing, the cluster incorporating the presented enhancements as described in Section 4 is still in production and is expected to become available in October 2009. Hence, we now provide first estimates and projections concerning the expected performance of the new cluster system. We will revise our figures as soon as the system (and adapted cryptanalytical implementations) become available. With the design modifications on the COPACOBANA architecture, we can firstly make use of more logic resources due to the larger Spartan-3 5000 FPGAs. With respect to the original Spartan-3 1000 FPGAs, the amount of logic has increased by a factor of 4.5 per FPGA device. For our exhaustive key search applications as shown in Section 3.1, we thus assume a linear speedup (at least) by a factor of 4. More precisely, the original DES breaking application implemented four engines per Spartan-3 1000 so that we expect 16 engines to run at the same clock frequency per Spartan-3 5000. This will reduce the average runtime to break DES to a single day and 19 hours with only one enhanced COPACOBANA. Similar linear performance speed-ups can be gained for the ePass cracker and A5/1 breaker (about 4.5 seconds and 1.5 hours, respectively). However, note that due to the higher cost of the enhanced COPACOBANA (which grows by a factor of 4.5 as well), the cost-performance is not better than with the original machine (even worse, when taking Moore's Law into account). In general, brute-force techniques do not benefit from the new architectural improvements, i.e., instead of one new COPACOBANA also five original machines based on Spartan-3 1000 can be used. In this case, the only advantage of the new design is due to the reduced power consumption.

The real advantages of the enhanced machine manifest for advanced cryptanalytical application with higher demands on the infrastructures such as communication throughput and local memories adjacent to the FPGAs. The efficient generation of TMTO tables will now become available so that we expect to finish the tables for A5/1 in less than a month. Furthermore, we are working towards an implementation of our ECM core (cf. Section 3.2.2) for Spartan-3A DSP 3400 FPGAs which also integrate 126 DSP slices, however at much lower costs compared to Virtex-4 devices. Finally, we plan to adapt the same implementation strategy based on DSP slices also for the Pollard-Rho ALU. This can also result in a gain in performance by an order of magnitude. Last but not least, the new COPACOBANA also could provide a suitable platform for even more complex applications, e.g., additional tasks required by the number field sieve, index calculus methods or lattice basis reduction algorithms.

6 Conclusion

In this work, we presented a series of cryptanalytical applications for a cost-efficient hardware architecture (COPACOBANA). According to our findings based on a varietly of cryptanalytical implementations, we identified shortcomings in the design of our cluster. Then, we came up with an enhanced version which also provides larger and more powerful FPGAs, up to 4 GB of local memory and fast point-to-point serial communication links between all devices and the controller. These modifications bring COPACOBANA into a promising position to tackle even more complex tasks in cryptanalsis as well as from general-purpose computing. With all these enhancements, in particular for communication and local memory, the new COPACOBANA cluster becomes even useful beyond pure code-breaking, catching up with respect to other supercomputing platforms, still at low costs.

References

[1] S. Babbage. A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers. In *European Convention on Security and Detection*, volume 408 of *IEE Conference Publication*, 1995.

[2] A. Biryukov and A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In *Proc. of ASIACRYPT'00*, volume 1976 of *LNCS*, pages 1–13. Springer, 2000.

[3] A. Biryukov, A. Shamir, and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC. In *Proc. of FSE'00*, volume 1978 of *LNCS*, pages 1–18. Springer, 2001.

[4] D. Carluccio, K. Lemke-Rust, C. Paar, and A.-R. Sadeghi. E-Passport: The Global Traceability or How to Feel Like an UPS Package. In *Proc. of WISA'06*, volume 4298 of *LNCS*, pages 391–404. Springer.

[5] Certicom Corporation. Certicom ECC Challenges, 2005. `http://www.certicom.com`.

[6] CESYS GmbH. USB2FPGA Product Overview. `http://www.cesys.com`, January 2005.

[7] D. Denning. Cryptography and Data Security. Addison-Wesley, 1982.

[8] G. de Meulenaer, F. Gosset, M. M. de Dormale, and J.-J. Quisqater. Integer Factorization Based on Elliptic Curve Method: Towards Better Exploitation of Reconfigurable Hardware. In *Proc. of FCCM'07*, pages 197–206. IEEE Computer Society, 2007.

[9] T. Finke and H. Kelter. Radio Frequency Identification – Abhörmöglichkeiten der Kommunikation zwischen Lesegerät und Transponder am Beispiel eines ISO14443-Systems. `http://www.bsi.de/fachthem/rfid/Abh_RFID.pdf`.

[10] K. Gaj, S. Kwon, P Baier, P. Kohlbrenner, H. Le, M. Khaleeluddin, and R. Bachimanchi. Implementing the Elliptic Curve Method of Factoring in Reconfigurable Hardware. In *Proc. of CHES '06*, volume 4249 of *LNCS*, pages 119–133. Springer, 2006.

[11] Timo Gendrullis, Martin Novotný, and Andy Rupp. A real-world attack breaking A5/1 within hours. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Proc. of CHES '08*, volume 5154 of *Lecture Notes in Computer Science*, pages 266–282. Springer, 2008.

[12] T. Güneysu, C. Paar, and J. Pelzl. Attacking Elliptic Curve Cryptosystems with Special-Purpose Hardware. In *Proc. of FPGA'07*, pages 207–215. ACM Press, 2007.

[13] Tim Güneysu, Timo Kasper, Martin Novotný, Christof Paar, and Andy Rupp. Cryptanalysis with COPACOBANA. *IEEE Trans. Comput.*, 57(11):1498–1513, November 2008.

[14] G. P. Hancke. Practical Attacks on Proximity Identification Systems (Short Paper). In *Proc. of SP'06*, pages 328–333. IEEE Computer Society, 2006.

[15] D. R. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Verlag, 2004.

[16] M. E. Hellman. A Cryptanalytic Time-Memory Trade-Off. In *IEEE Transactions on Information Theory*, volume 26, pages 401–406, 1980.

[17] J.-H Hoepman, E. Hubbers, B. Jacobs, M. Oostdijk, and R. Wichers Schreur. Crossing Borders: Security and Privacy Issues of the European e-Passport. In *Proc. of IWSEC'06*, volume 4266 of *LNCS*, pages 152–167. Springer, 2006.

[18] ISO/IEC 14443. Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part 1-4. www.iso.ch, 2001.

[19] S. Vaudenay J. Monnerat and M. Vuagnoux. About Machine-Readable Travel Documents. In *Proc. of RFIDSec'07*, pages 15–28, 2007.

[20] A. Juels, D. Molnar, and D. Wagner. Security and Privacy Issues in E-Passports. In *Proc. of SecureComm'05*, pages 74–88. IEEE Computer Society, 2005.

[21] T. Kasper, D. Carluccio, and C. Paar. An Embedded System for Practical Security Analysis of Contactless Smartcards. In *Proc. of WISTP'07*, volume 4462 of *LNCS*, pages 150–160. Springer, 2007.

[22] 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.

[23] J. Keller and B. Seitz. A Hardware-Based Attack on the A5/1 Stream Cipher. Technical report, Fernuni Hagen, Germany, 2001. `http://pv.fernuni-hagen.de/docs/apc2001-final.pdf`.

[24] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler. How to Break DES for € 8,980. In *SHARCS Workshop*. Cologne, Germany, 2006.

[25] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker. In *Proc. of CHES '06*, volume 4249 of *LNCS*, pages 101–118. Springer, 2006.

[26] H. Lenstra. Factoring Integers with Elliptic Curves. *Annals Math.*, 126:649–673, 1987.

[27] Y. Liu, T. Kasper, K. Lemke-Rust, and C. Paar. E-Passport: Cracking Basic Access Control Keys. In *Proc. of OTM'07, Part II*, volume 4804 of *LNCS*, pages 1531–1547. Springer, 2007.

[28] N. Mentens, L. Batina, B. Prenel, and I. Verbauwhede. Time-Memory Trade-Off Attack on FPGA Platforms: UNIX Password Cracking. In *Proc. of ARC'06*, volume 3985 of *LNCS*, pages 323–334. Springer, 2006.

[29] ICAO TAG MRTD/NTWG. Biometrics Deployment of Machine Readable Travel Documents, Technical Report, 2004.

[30] NIST FIPS PUB 46-3. *Data Encryption Standard*. Federal Information Processing Standards, National Bureau of Standards, U.S. Department of Commerce, January 1977.

[31] P. Oechslin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *Proc. of CRYPTO'03*, volume 2729 of *LNCS*, pages 617–630. Springer, 2003.

[32] National Institute of Standards and Technology. FIPS 180-3 Secure Hash Standard (Draft). `http://www.csrc.nist.gov/publications/PubsFIPS.html`.

[33] J. M. Pollard. Monte Carlo Methods for Index Computation mod p. *Mathematics of Computation*, 32(143):918–924, July 1978.

[34] 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.

[35] Harko Robroch. ePassport Privacy Attack, Presentation at Cards Asia Singapore, April 26,2006. `http://www.riscure.com`.

[36] 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 *Proc. of FPGA'03*, pages 247–247. ACM, 2003.

[37] M. Šimka, J. Pelzl, T. Kleinjung, J. Franke, C. Priplata, C. Stahlke, M. Drutarovský, V. Fischer, and C. Paar. Hardware Factorization Based on Elliptic Curve Method. In *Proc. of FCCM'05*, pages 107–116. IEEE Computer Society, 2005.

[38] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat. A Time-Memory Tradeoff using Distinguished Points: New Analysis & FPGA Results. In *Proc. of CHES '02*, volume 2523 of *LNCS*, pages 596–611. Springer, 2002.

[39] P.C. van Oorschot and M.J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12(1):1–28, 1999.

[40] Xilinx. Spartan-3 FPGA Family: Complete Data Sheet, DS099. `http://www.xilinx.com`, January 2005.