Exact Cost Estimates for ECC Attacks with Special-Purpose Hardware

Jan Pelzl
Chair for Communication Security
Horst Görtz Institute for IT-Security

10th Workshop on Elliptic Curve Cryptography, September 18-20, 2006, Toronto

Acknowledgement

Special thanks to

Tim Güneysu and Christof Paar
(Horst Görtz Institute, University of Bochum)

Agenda

Horst-Görtz Institut für IT Sicherheit

- Introduction
- How to Solve Elliptic Curve Discrete Logarithms?
- Special Purpose-Hardware
- A Hardware Architecture for Pollard‘s Rho
- Results of the FPGA Implementation and Extrapolation
- Conclusion

Introduction

Horst-Görtz Institut
für IT Sicherheit

Short quiz: why are we here?

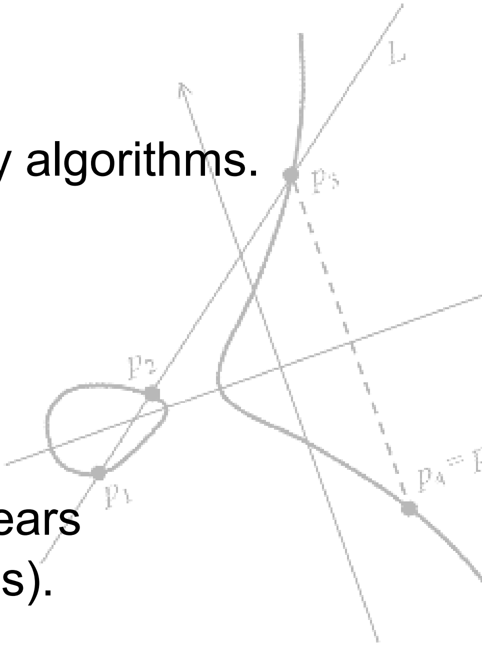
- ☐ I do not know, my boss sent me.
- ☐ Elliptic curve cryptography is superflous - I just want to spend some nice days in Toronto…
- ☒ Elliptic curve cryptography gains in importance in applications.

(**AND** we want spend some nice days in Toronto…)

Introduction

As we all know…

- ECC can be **more efficient** than (most) other public-key algorithms.
- In general: **only generic attacks** known to break ECC.
- ECC has become more **wide-spread** over the last 10 years (partially driven by an increase in embedded applications).
- Trend: ECC over **GF(*p*) more popular** over GF(2^m).

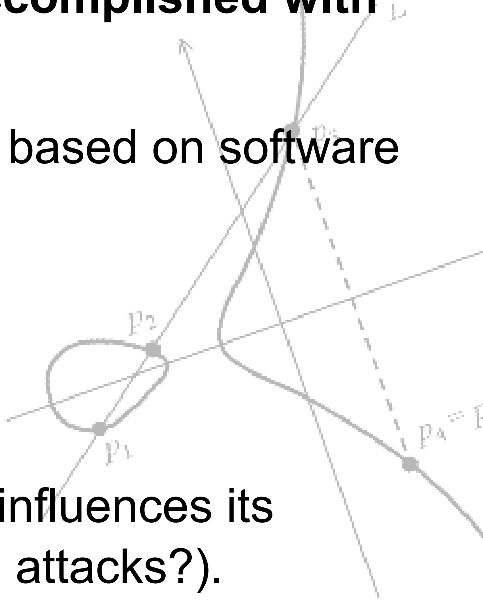


Horst-Görtz Institut
für IT Sicherheit

Introduction

Facts:

- All previous attacks (e.g., Certicom challenges) were **accomplished with software** implementations.
- It is very unlikely that future attacks against ECC will be based on software (**hardware is more cost-effective**).



But we (still) do not know…

- … how far special-purpose hardware for breaking ECC influences its security (are 160 bits really sufficient against HW-based attacks?).
- … what the overall costs of a generic attack against ECC in hardware are.

Introduction

Security of ciphers is related to complexity of attacks:

- **Symmetric ciphers:**
  - usually, only exhaustive key search possible (brute force)
  - an exhaustive key search should be infeasible in practice
  - **common key lengths: 112…256 bits**
  - „> 80 bits are safe“
- **Asymmetric ciphers** (RSA, ElGamal, …):
  - larger keys due to index calculus
  - **common key lengths: 1024…4096 bits**
  - limit of software-based attacks: 768 bits (?)
  - „> 1024 bits are safe“
- **Asymmetric ciphers** (ECC):
  - Only generic attacks possible
  - **common key lengths: 160…256 bits**
  - „> 160 bits are safe“

Introduction

Role of hardware for code-breaking:

- **Well analyzed for several „weak“ symmetric ciphers** such as, e.g. DES:
  - Deep Crack (ASIC cluster) [1]
  - COPACOBANA (FPGA cluster) [2,16]
- Current **(strong) symmetric ciphers are out of reach** (AES, etc.)
  Exceptions (= virtually existent in practice):
  - Badly chosen passwords
  - Implementational flaws such as weak key derivation functions
  - Future progress in cryptanalysis (cf. MD5, SHA-1, …)
- **Quite well analyzed for asymmetric primitives such as RSA**
  - TWINKLE, TWIRL, YASD, SHARK, … [3-6]
  - But: feasibility of such complex designs is questionable
- **Hardly analyzed for ECC** (no proof-of-concept till 2006)
  - First estimate by Oorschot/Wiener in 1999 (paper & pencil) [7]
  - First proof-of-concept implementations of Pollard‘s Rho in 2006:
    o Güneysu/Paar/Pelzl for GF(*p*) [8]
    o Bulens/Meurice/Quisquater for GF(2^m) [9]

Introduction

Big question: how secure is ECC against hardware-based attacks?

- Optimal plattform for cryptanalysis of ECC?
- Alike security of ECC over GF(*p*) and GF(2^m)?
- Comparison to software-based attacks
- Comparison to other asymmetric ciphers

Agenda

- Introduction
- How to Solve Elliptic Curve Discrete Logarithms?
- Special Purpose-Hardware
- A Hardware Architecture for Pollard‘s Rho
- Results of the FPGA Implementation and Extrapolation
- Conclusion

How to Solve ECDLPs

A cryptographic primitive of ECC used in many protocols is the Elliptic Curve Discrete Logarithm Problem (ECDLP)

- Let *P* be a point on an elliptic curve

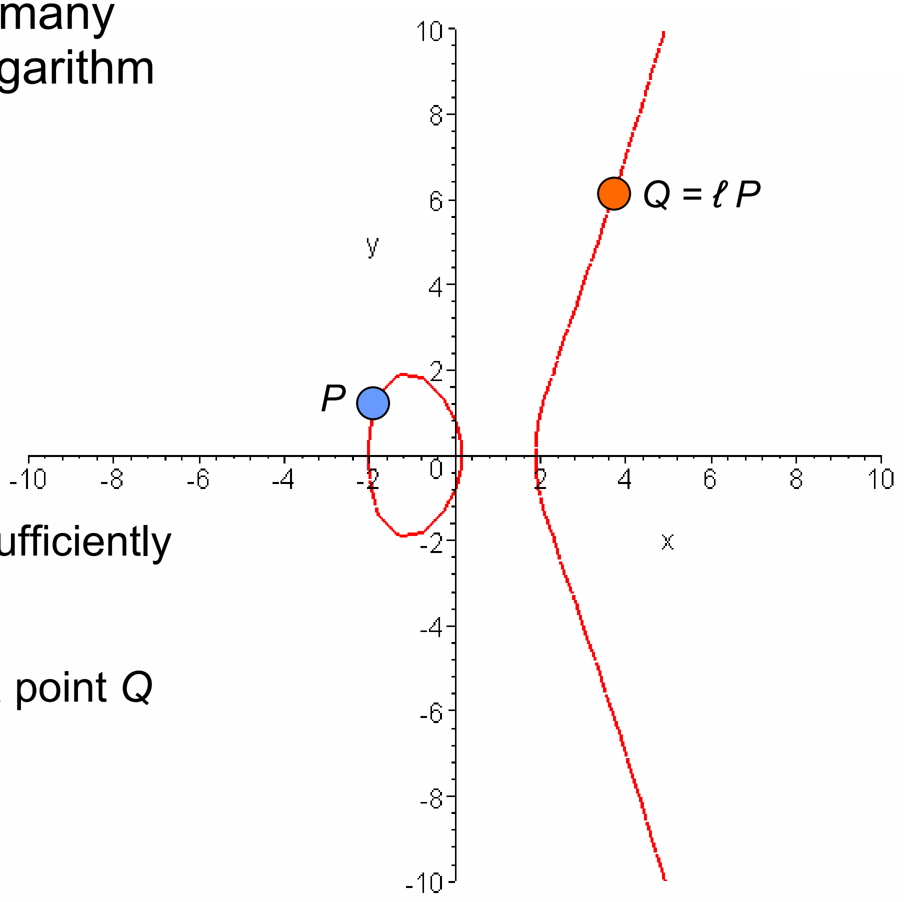
  $E: \; y^2=x^3+ax+b$ over a field K

  with point order $n = ord(P)$.

- Furthermore, let *P* be a generator of a sufficiently large subgroup.

  Determine the **discrete logarithm** ℓ of a point *Q such that*

  $$Q = \ell P.$$



How to Solve ECDLPs

Known (generic) methods to solve the ECDLP

- **Naïve Search**: Sequentially test *P*, *2P, 3P, 4P*,…
  - Brute force attack is infeasible for groups with more than 2^{80} elements
- **Shank's Baby-Step-Giant-Step Method** [10]
  - Complexity in time AND memory of about $\sqrt{n}$
- **Pollard's Lambda method** [11]
  - Efficient method for bounded search within an interval $1<b<n$
  - Complexity dependent on bound b with $3.28 \sqrt{b}$
- **Pollard's Rho method** [12]
  - Most efficient algorithm for solving general ECDLP known so far
  - Parallel implementation possible
  - Complexity of $\sqrt{(\pi n / 2)}$

Note: All attacks have **exponential** complexity

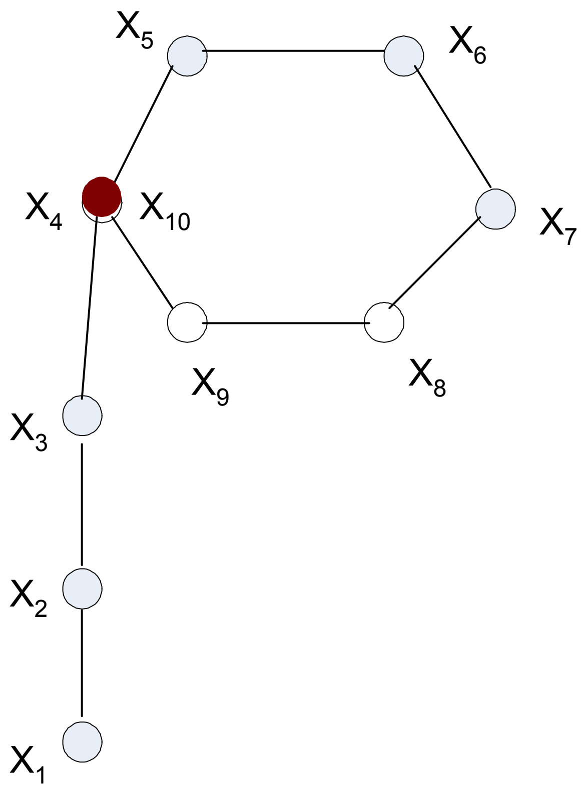
How to Solve ECDLPs

Known methods to solve the ECDLP on special (weak) EC over GF(p) with subexponential complexity [13]:

- Supersingular curves
- Anomalous curves (Curves over GF(p) with exactly p points)
  (Attack by Araki-Satoh-Semaev-Smart)
- Curves vulnerable to Weil and Tate Pairing attacks
  (Attack in polynomial time when $n \mid q^k-1$ for small k)

How to Solve ECDLPs

Single Processor Pollard Rho (SPPR)



Collision path of pseudo-random walk

SPPR originally proposed by J. Pollard in 1978 [12]

Idea: Find a collision of two arbitrary points X_k, X_l while monitoring their relative distance to P and Q via

$$c_k P + d_k Q = X_k = X_l = c_l P + d_l Q.$$

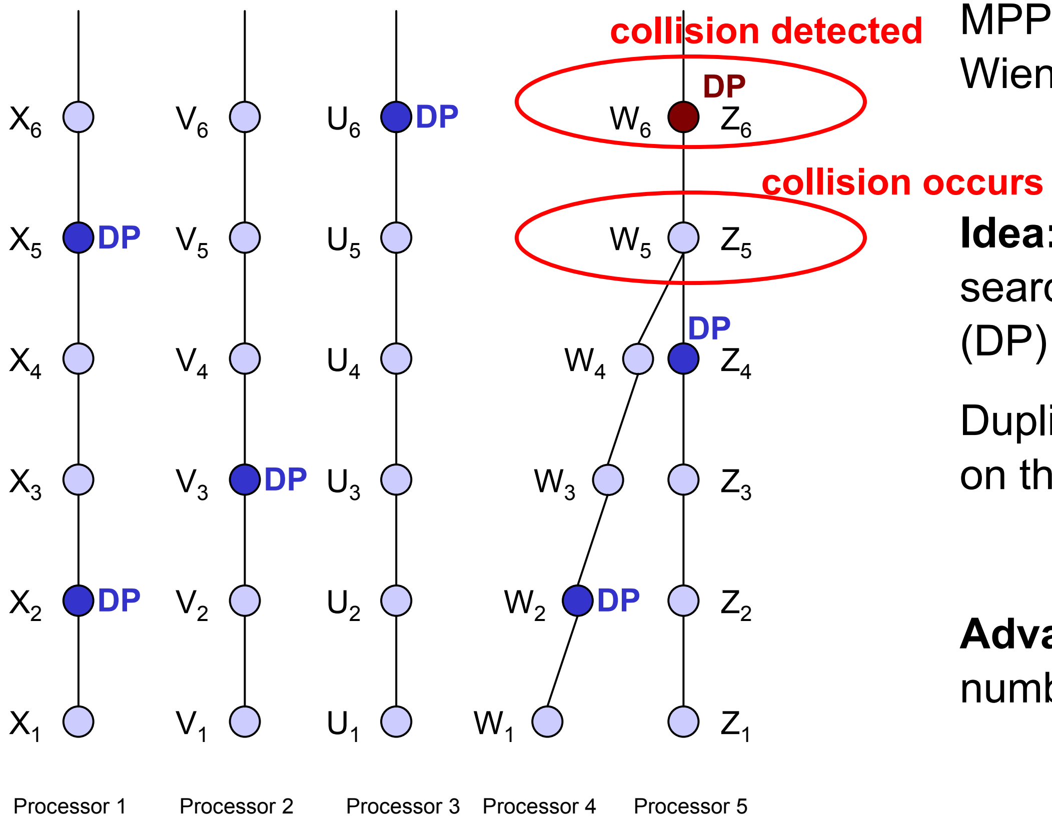
Then, the ECDLP is given by

$$\ell = (c_k - c_l)\,(d_k - d_l)^{-1} \bmod n$$

Collisions are detected with Floyd's cycle-finding algorithm using a pseudo random walk

How to Solve ECDLPs

Multi Processor Pollard Rho (MPPR)



collision detected
DP
collision occurs
X_6 X_5 X_4 X_3 X_2 X_1
V_6 V_5 V_4 V_3 V_2 V_1
U_6 U_5 U_4 U_3 U_2 U_1
W_6 W_5 W_4 W_3 W_2 W_1
Z_6 Z_5 Z_4 Z_3 Z_2 Z_1
DP
Processor 1
Processor 2
Processor 3
Processor 4
Processor 5

Colliding DP trails of multiple processors w_i

MPPR proposed by van Oorschot and Wiener in 1999 [7]

Idea: Multiple processors have individual search paths for "Distinguished Points" (DP) which are sent to a central server

Duplicate distinguished points detected on the server reveal ECDLP

Advantage: Linear speed-up with number of employed processors

How to Solve ECDLPs

Notion of a „distinguished point“ (DP)

- Subset of the set of all points
  - Should occur “not too seldom” and “not too often” (trade-off)
  - Optimum ratio depends on implementational aspects

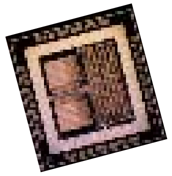
- “Easy” to distinguish
  - Fast evaluation of distinguished property
  - Often used distinguished property:
    “least significant k bits of x-coordinate are zero”, $k \sim 30$
  - Problem with projective space: point notation not unique...

How to Solve ECDLPs

Implementational issues

- GF(p) is **faster** than GF(2^m) **in software**
  - can use integer arithmetic units (e.g., Pentium's fast 32x32 bit multipliers)
  - GF(2^m) arithmetic (multiplication) not supported by standard CPUs

- GF(2^m) is **more efficient** than GF(p) **in hardware**
  - Arithmetic over GF(2^m) can be implemented very efficiently
  - GF(p) arithmetic more costly in area



How to Solve ECDLPs

State-of-the-art in ECC-Attacks

Certicom challenges for ECC over GF(p) and GF(2^m) [14]

Curve	Field size (bits)	Machine days*	Status
ECC2-79	79	352	Solved (12/1997)
ECCp-79		146	Solved (12/1997)
ECC2-97	89	180448	Solved (3/1998)
ECCp-97		71982	Solved (9/1998)
ECC2-109	109	$2.1 \cdot 10^7$	Solved (4/2004)
ECCp-109		$9 \cdot 10^6$	Solved (11/2002)
ECC2-131	131	$6.6 \cdot 10^{10}$	-
ECCp-131		$2.3 \cdot 10^{10}$	-
ECC2-163	163	$2.9 \cdot 10^{15}$	-
ECCp-163		$2.3 \cdot 10^{15}$	-

* based on a Pentium 100

How to Solve ECDLPs

ECC Attacks: Status Quo

The 109-bit challenges have been solved by Pollard-Rho clusters:

- *ECCp-109 solved in Nov. 2002*
- *ECC2-109 solved in April 2004*

For ECCp-109, it took 10,000 computers (mostly PCs) running 24 hours a day for 549 days!

E.g., 163-bit challenge (ECC2-163 or ECCp-163) is **10^7-10^8 times more complex**

► out of reach for software-based attacks

► more cost-effective: use special-purpose hardware

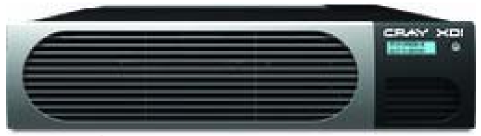
Agenda

- Introduction
- How to Solve Elliptic Curve Discrete Logarithms?
- Special Purpose-Hardware
- A Hardware Architecture for Pollard‘s Rho
- Results of the FPGA Implementation and Extrapolation
- Conclusion

Special-Purpose Hardware

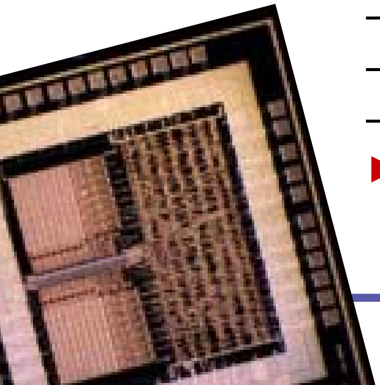
Possible solutions to computationally extensive problems:

- Large supercomputers:
  - Complex and expensive parallel computing architectures
  - Fast I/O, large memory, easy to program
  - E.g., Cray-XD1
  - ► Too complex for (most) cryptanalysis (bad cost-performance ratio)



- Distributed computing (conventional PCs):
  - Dedicated clients in clusters, or
  - Using PC‘s idle time: E.g., SETI@home (BOINC framework)
  - ► Problem of motivating for cryptanalytic challenges, confidentiality issues

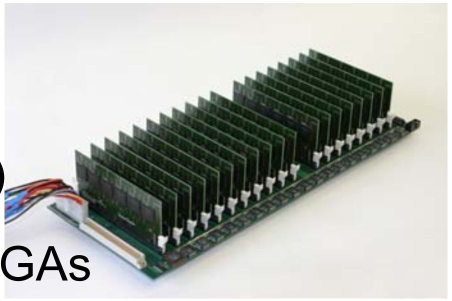
- Special-purpose hardware:
  - Application Specific Integrated Circuits (ASICs, high NRE)
  - Field Programmable Gate Arrays (FPGAs, low NRE)
  - Optimized for one particular objective
  - ► Tradeoff between reprogrammability and price per piece, best cost-performance ratio



Special-Purpose Hardware

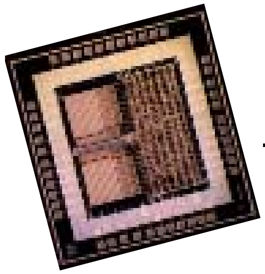
Platform costs:

Software based architecture (Pentium M@1.7GHz)
– Costs: including overhead ≈ US$ 400



FPGA based architecture (Xilinx XC3S1000; 10^6 equ. gates)

- Costs: based on COPACOBANA ≈ US$10,000 per 120 FPGAs



Estimated ASIC based architecture ($10x10^6$ transistors @ 500MHz)
– Costs: including overhead ≈ US$50 (excluding NRE)

Example: for US$10,000,000 we get 25,000 Pentiums, or 120,000 FPGAs, or 200,000 ASICs

Special-Purpose Hardware

Common design methodology

- Development of the architecture

↓

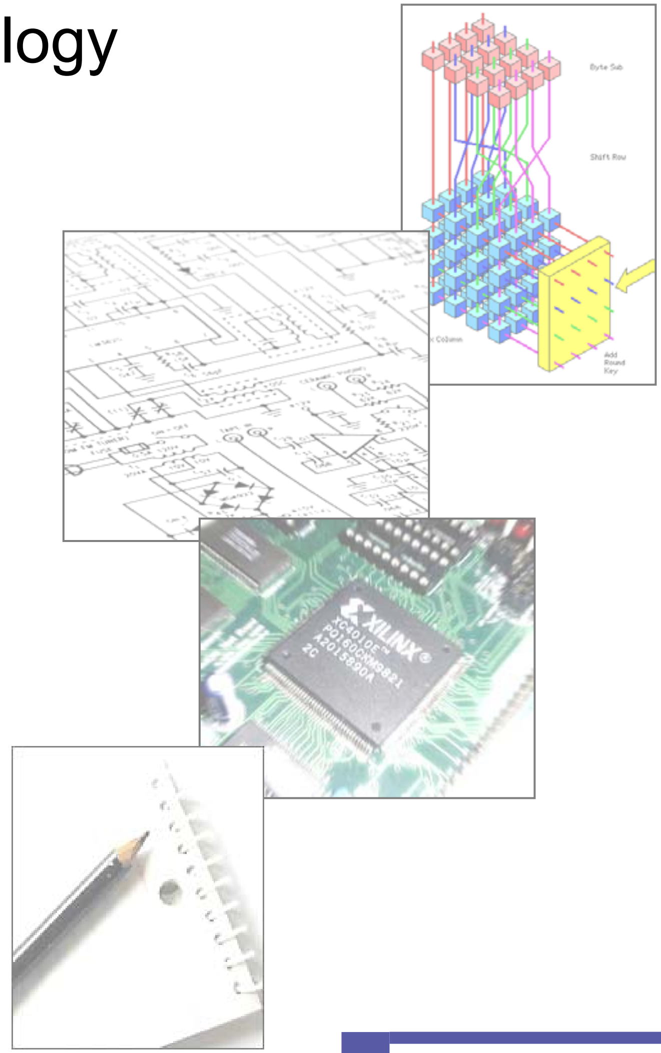
- Implementation in hardware description language (VHDL)

↓

- Run code on programmable hardware (FPGA) as proof-of–concept

↓

- Use running FPGA implementations for further (fairly accurate) estimates



Special-Purpose Hardware

Possible metrics for a „good“ design:

- **Time:** make design as fast as possible
  (loop unrolling, pipelining, parallel ALUs, table look-ups …)
- **Area:** make design as small as possible
  (serialization, no table look-ups, …)
- **Area-Time (AT) product:** minimize the product of area and execution time

► **AT-optimized** architectures are **most cost-effective!**
(Lowest cost per computation)

Agenda

- Introduction
- How to Solve Elliptic Curve Discrete Logarithms?
- Special Purpose-Hardware
- A Hardware Architecture for Pollard‘s Rho
- Results of the FPGA Implementation and Extrapolation
- Conclusion

Parallel Pollard‘s Rho in Hardware

Remarks:

- Focus on generic curves defined over GF(*p*)
  - General case gives upper bound on complexity of attacks
  - Mostly used in practice, especially in software
  - For GF(2^m): estimates given by Bulens et al. [9]
- Use hardware to accelerate time critical operations
  - Implement search for distinguished points in hardware (point processors)
  - Collect DPs on a central server (e.g., a simple PC)
- Cost-efficient design of point processors
  - AT-minimized (= cost-effective) arithmetic units
  - Low memory usage in hardware

Parallel Pollard‘s Rho in Hardware

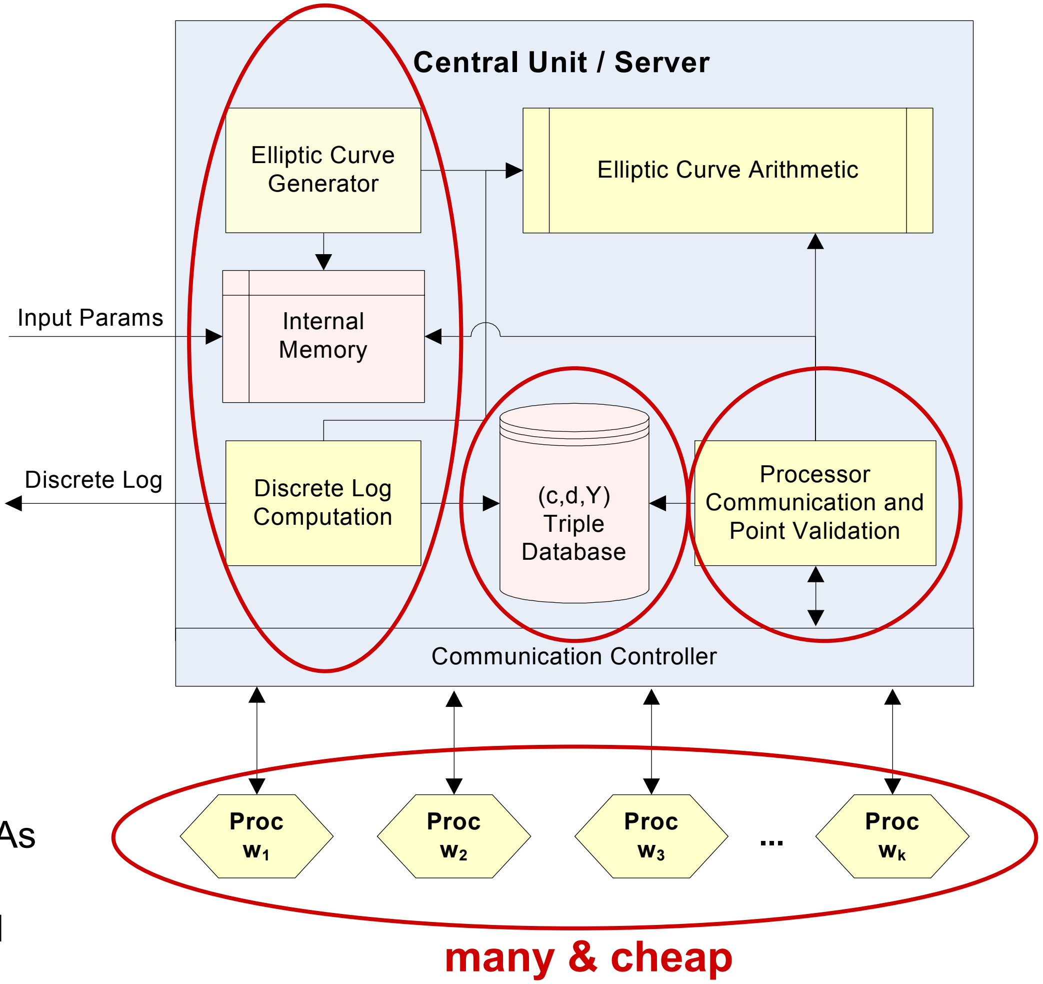
Overview:

Central server (software-based)

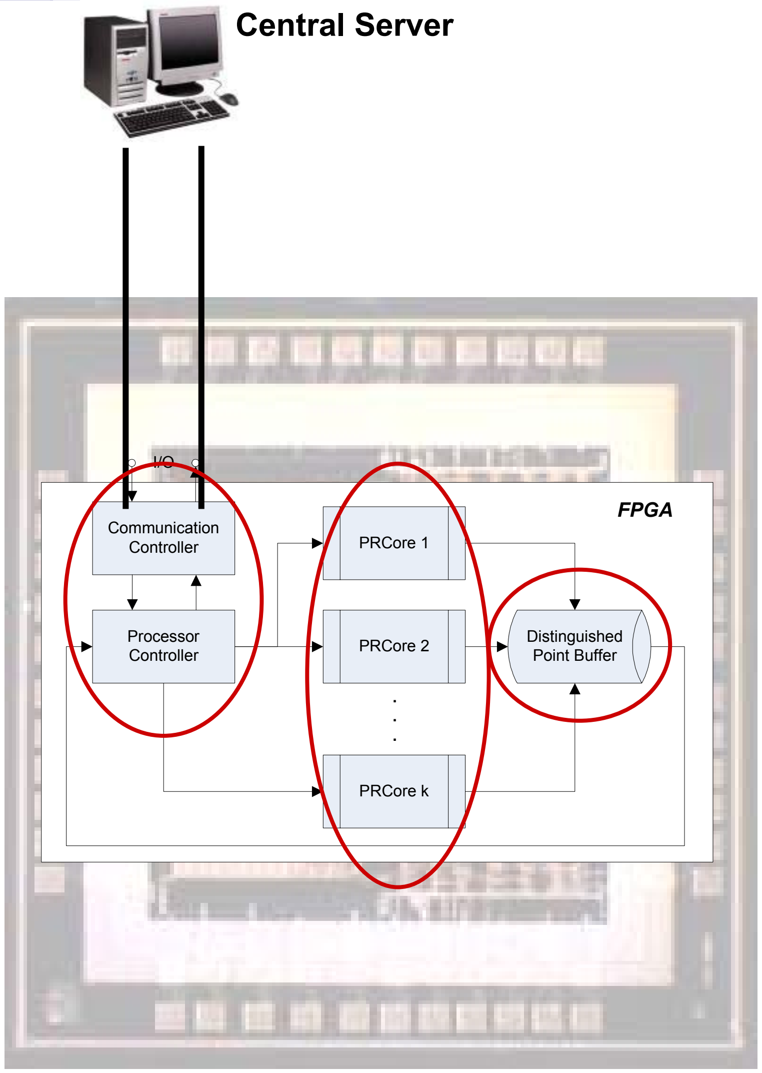
- **Administrative** tasks
- Centralized DP **database**
- **Manages** attached point processors w_i

Point processors w_i (hardware-based)

- **Compute distinguished points** and transfer them to server
- Implemented as an large array of FPGAs or ASICs
- FPGAs offer more design flexibility and will be used for a first implementation



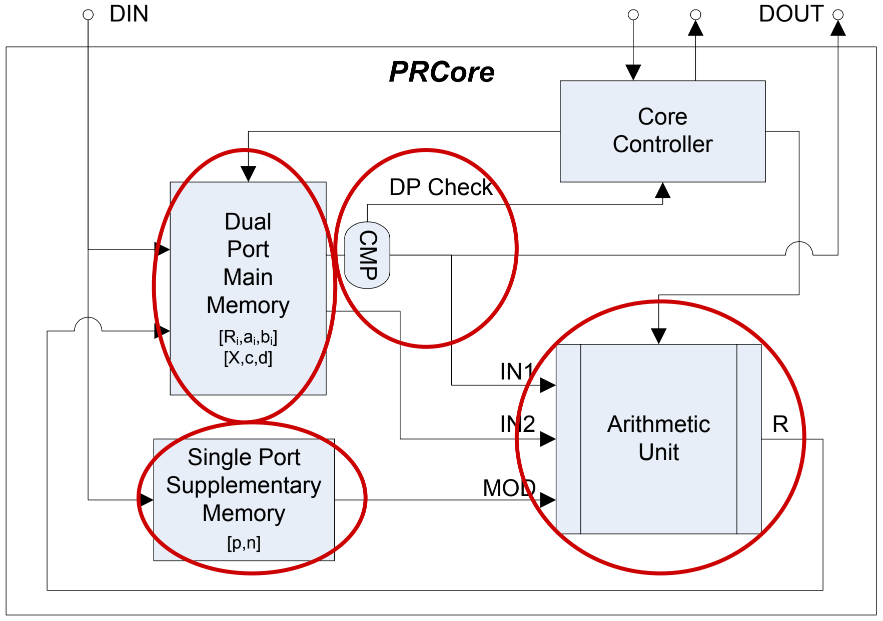
Parallel Pollard‘s Rho in Hardware



Top level design (chip):

- Each FPGA: **multiple point engines** (*PRCore*) to compute separate trails.
- All cores store distinguished points in a **shared point buffer**.
- Buffer locking & **host communication** are needed to transfer DPs to the server.

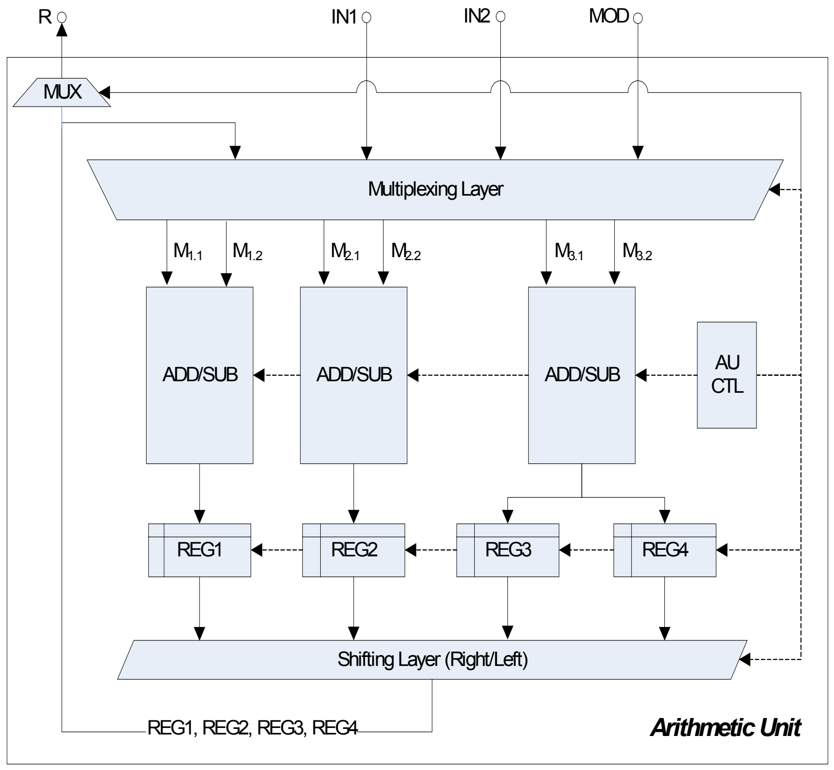
Parallel Pollard‘s Rho in Hardware



Core level design:

- Each core has an **Arithmetic Unit** (AU) for modular computations [15].
- Storage for **current point** X_i and coefficients c_i, d_i with $X_i = c_iP + d_iQ$
- 16 **random points** $R_1 \ldots R_{16}$
- Pseudo-random walk
  $$X_{i+1} = X_i + R_\theta$$
- Distinguished point **detection unit** (comparison if m LSBs are zero)

Parallel Pollard‘s Rho in Hardware



AU level design:

- ECC computations use **affine coordinates** to preserve a simple DP property.
- Modular operations: addition, subtraction, multiplication and inversion
- AU uses **Montgomery representation** for efficient modular arithmetic:
  - Montgomery multiplication
  - modified Kaliski inversion algorithm
- Search for DPs is performed completely in Montgomery domain.

Agenda

- Introduction
- How to Solve Elliptic Curve Discrete Logarithms?
- Special Purpose-Hardware
- A Hardware Architecture for Pollard‘s Rho
- Results of the FPGA Implementation and Extrapolation
- Conclusion

Results and Extrapolation

Point throughput on an FPGA:

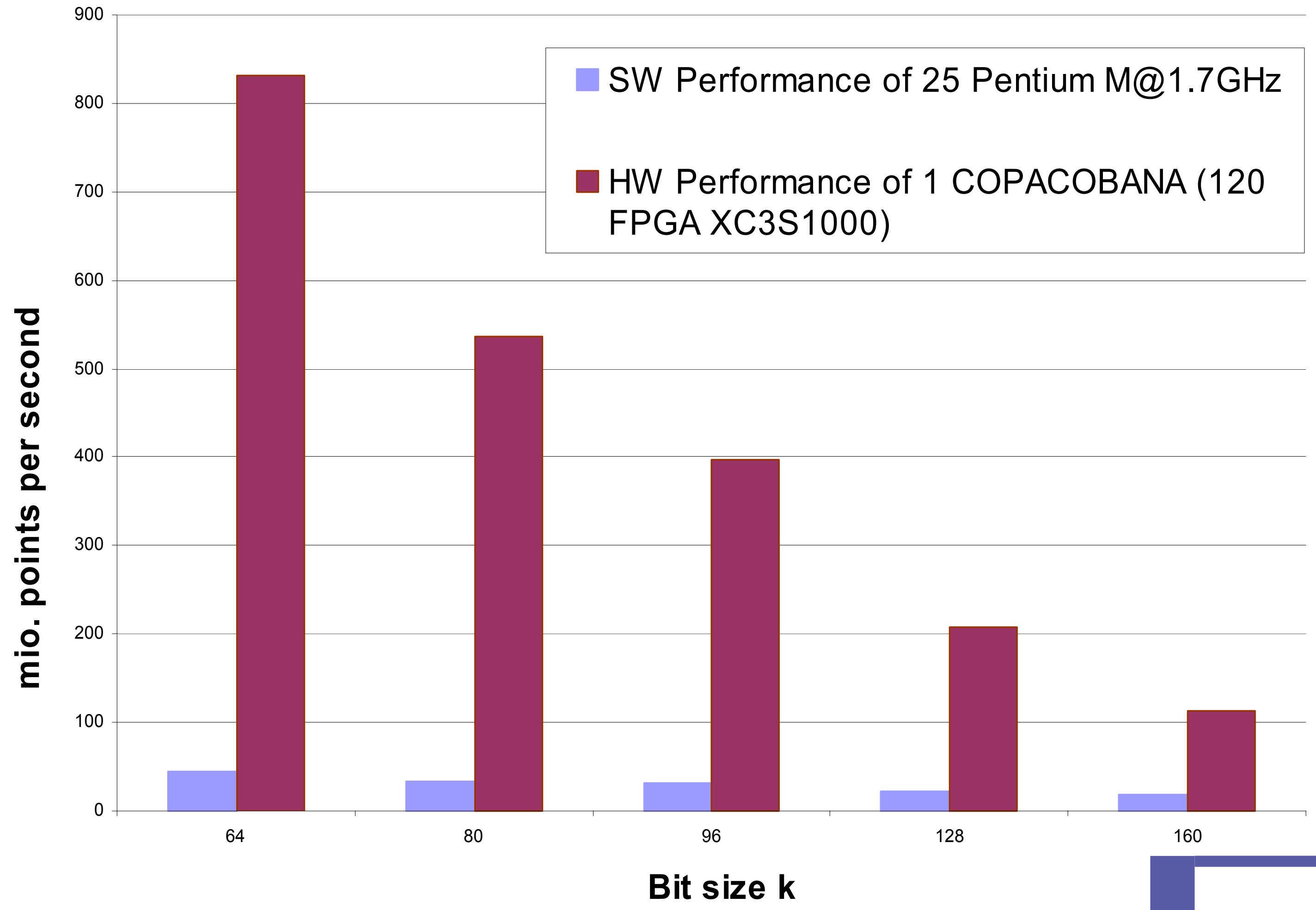
Performance results for GF(*p*): Pollard-Rho architecture synthesized on a Spartan3-1000 FPGA [8]

Bit size k	# Cores	Device Usage	Max Freq.	Time per Operation	Pts/sec per Core	Pts/sec per FPGA
160	2	83 %	40.0 MHz	21.4 µs	46,800	93,600
128	3	98 %	40.1 MHz	17.3 µs	57,800	173,000
96	4	98 %	44.3 MHz	12.1 µs	82,700	331,000
80	4	88 %	50.9 MHz	8.94 µs	111,900	447,000
64	5	88 %	52.0 MHz	7.21 µs	138,600	693,000

Results and Extrapolation

Horst-Görtz Institut
für IT Sicherheit

Comparison for GF(*p*): Software and FPGA-Hardware for $US 10,000



Results and Extrapolation

What can we achieve with $US 1,000,000?

Expected runtime of a successful attack (GF(*p*)) depending on bit size *k* [8]

Bit size k	SW Reference Pentium M@1.7	Implementation XC3S1000 FPGA	Estimated ASIC Performance
80	40.6 h	2.58 h	-
96	8.04 d	14.8 h	-
112*	6.48 y	262 d	1.29 d
128	1.94×10^3 y	213 y	1.03 y
160	1.51×10^8 y	2.58×10^7 y	1.24×10^5 y

* SEC-1 specified by SECG (Standards for Efficient Cryptography)

Results and Extrapolation

What can we achieve with even more funding?

Expected runtime of successful attack (GF(*p*)) on *k*-bit curves for different funding (ASIC) [8]

k	**US$ 10^5**	**US$ 10^6**	**US$ 10^7**	**US$ 10^8**
128	1.03×10^1 y	1.03 y	0.103 y	0.0103 y
160	1.24×10^6 y	1.24×10^5 y	1.24×10^4 y	1.24×10^3 y
192	9.64×10^{10} y	9.64×10^9 y	9.64×10^8 y	9.64×10^7 y
256	1.09×10^{21} y	1.09×10^{20} y	1.09×10^{19} y	1.09×10^{18} y

Results and Extrapolation

Estimates: Attacks on ECC standards

Average duration of successful Pollard Rho attack on a single system [8]

Challenge/ Standard	Est. time to solve*	SW Reference Pentium M@1.7	Implementation XC3S1000 FPGA	Estimated ASIC Performance
Cost per chip inc. overhead:		**$US 400**	**$US 83**	**$US 50**
ECCp-79	146 d	49.0 d	15.3 d	-
ECCp-97	71982 d	74.7 y	30.7 y	-
ECCp-109	9.0×10^{6} d	5.57×10^{3} y	2.91×10^{3} y	-
SEC-1 (112 bit)	-	1.62×10^{4} y	8.64×10^{3} y	-
ECCp-131	2.3×10^{11} d	1.40×10^{7} y	7.40×10^{6} y	9.34×10^{4} y
ECCp-163	2.3×10^{15} d	1.09×10^{12} y	9.15×10^{11} y	1.16×10^{10} y
ECCp-191	4.8×10^{19} d	2.17×10^{16} y	1.89×10^{16} y	2.39×10^{14} y
ECCp-239	1.4×10^{27} d	4.44×10^{23} y	8.62×10^{23} y	1.01×10^{22} y

Agenda

- Introduction

- How to Solve Elliptic Curve Discrete Logarithms?
- Special Purpose-Hardware
- A Hardware Architecture for Pollard‘s Rho
- Results of the FPGA Implementation and Extrapolation
- Conclusion

Conclusion

Estimated cost of a successful attack within 1 year

Expected cost of a successful attack in one year depending on cryptosystem

Cryptosystem	Cost in $US	Architecture (ASIC)
ECCp-131	$5 \cdot 10^6$	Güneysu/Paar/Pelzl [8]
ECC2-163	$7 \cdot 10^{10}$	Bulens/Meurice/Quisquater [9] 1)
ECCp-163	$6 \cdot 10^{11}$	Güneysu/Paar/Pelzl [8]
RSA-1024	$2 \cdot 10^8$	SHARK [5]
	10^7	TWIRL [4]

1) Based on the assumption that the architecture can be realized as ASIC for $US 100 including overhead

Conclusion

- First proof-of-concept implementations of parallel Pollard‘s rho attack for ECC over GF(p) and GF(2^m) available this year.
- Compared to GF(2^m), ECC over GF(p) is an order of magnitude harder to break with special-purpose hardware.
- ECC seems very secure with current attacks and technology, e.g., ASIC attack @ $US 5 mio. for ECCp-131 within one year.
- ECCp-163 attack within one year: \$US $6 \cdot 10^{11}$.
  - According to Moore‘s Law it will take about 20 years to perform the same attack for $US 1 mio.
- SEC-1 standard by SECG with 112 bits is insecure!
- Based on estimates for RSA-1024, ECCp-163 would be (at least) ~3000 times more expensive to break!

Conclusion

Future work and open problems

- Analysis on parallel FPGA cluster (COPACOBANA)
- Find efficient distinguished property in projective space (both for GF(p) and GF(2^m))
- Analysis and possible extension to hyperelliptic case (genus-2)
- Take part in challenges with FPGA cluster ☺

References

[1] Electronic Frontier Foundation (EFF), *Cracking {DES}: Secrets of Encryption Research*, Wiretap Politics & Chip Design, O'Reilly & Associates Inc., July, 1998, ISBN 1-56592-520-3.

[2] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, M. Schimmler, *Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker*, Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10 - October 13, 2006, Proceedings. LNCS, Springer-Verlag.

[3] A. Shamir, *Factoring Large Numbers with the Twinkle Device*, 1st International Workshop on Cryptographic Hardware and Embedded Systems - CHES 1999, Worcester, MA, USA, August, 1999, vol. 1717, LNCS, pp. 2-12, Springer-Verlag.

[4] A. Shamir, Tromer, *Factoring Large Numbers with the TWIRL Device*, Advances in Cryptology - CRYPTO'03, pp. 1--26, 2003, vol. 2729, LNCS, Springer-Verlag.

[6] J. Franke, T. Kleinjung, C. Paar, J. Pelzl, C. Priplata, and C. Stahlke, *SHARK - A Realizable Hardware Architecture for Factoring 1024-bit Composites with the GNFS*, Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. LNCS 3659, Springer-Verlag.

[5] W. Geiselmann and R. Steinwandt, *Yet Another Sieving Device*, Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, pp. 278-291, 2004,eds. Tatsuaki, Okamoto, vol. 2964, LNCS, February, Springer-Verlag.

[7] P.C. van Oorschot, M.J. Wiener, *Parallel Collision Search with Cryptanalytic Applications*, Journal of Cryptology, vol. 12, no. 1, pp.1-28, 1999.

[8] T.E. Gueneysu, C. Paar, J. Pelzl, *On the Security of Elliptic Curve Cryptosystems against Attacks with Special-Purpose Hardware*, 2nd Workshop on Special-purpose Hardware for Attacking Cryptographic Systems - SHARCS 2006, April 3-4, 2006, Cologne, Germany.

[9] Philippe Bulens, Guerric Meurice de Dormale and Jean-Jacques Quisquater, *Hardware for Collision Search on Elliptic Curve over GF(2m),* 2nd Workshop on Special-purpose Hardware for Attacking Cryptographic Systems - SHARCS 2006, April 3-4, 2006, Cologne, Germany.

References

[10] D. Shanks, *Class number, a theory of factorization and genera*, Proc. Symp Pure Math., 1971, vol. 20, pp. 415-440.

[11] R. Gallant, R. Lambert, S. Vanstone, *Improving the parallelized Pollard lambda search on anomalous binary curves*, Mathematics of Computation, AMS, 2000, vol. 69, pp. 1699-1705.

[12] J. M. Pollard, *Monte Carlo methods for index computation mod p*, Mathematics of Computation, 1978, no. 143, pp. 918-924, July, vol. 32.

[13] D. Hankerson, A. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, 2004, Professional Computing, February,ISBN 038795273.

[14] Certicom Corporation, *Certicom ECC Challenges*, 2005, available at http://www.certicom.com.

[15] J. Pelzl, "*Hardware Implementation of ECC*, Slides of the ECC Summer School 2004, September 16th, 2004, Ruhr University Bochum, Germany. Available at http://www.ruhr-uni-bochum.de/itsc/tanja/summerschool/talks/hardware.pdf.

[16] COPACOBANA – Special-purpose hardware for code-breaking. Website: http://www.copacobana.org.