Experimental Comparison of Crypto-processors Architectures for Elliptic and Hyper-Elliptic Curves Cryptography

Gabriel GALLIN, Arnaud TISSERAND and Nicolas VEYRAT-CHARVILLON

CNRS – IRISA – University Rennes 1 – CAIRN
HAH Project

13th CryptArchi Workshop: June 29-30th, 2015

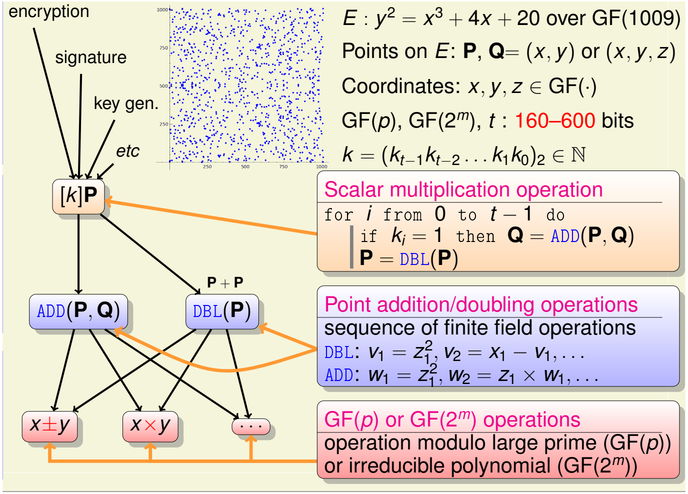
Summary

Summary

Asymmetric Cryptography: ECC and HECC

- Cryptographic primitives for protocols such as digital signature, exchange of secret keys and some specific encryption schemes
- Elliptic Curve Cryptography (ECC):
  - Actual standard for public key crypto-systems
  - Better performance and lower cost than RSA
- Hyper-Elliptic Curve Cryptography (HECC):
  - Evolution of ECC focusing a larger set of curves
  - Studied for future generations of asymmetric crypto-systems

Operations for (H)ECC



$E : y^2 = x^3 + 4x + 20$ over GF(1009)

Points on E: $\mathbf{P}$, $\mathbf{Q}$ $= (x, y)$ or (x, y, z)

Coordinates: $x, y, z \in \mathrm{GF}(\cdot)$

GF(p), GF(2^m), t : 160–600 bits

$k = (k_{t-1}k_{t-2}\dots k_1k_0)_2 \in \mathbb{N}$

Scalar multiplication operation

```
for $i$ from $0$ to $t-1$ do
    if $k_i = 1$ then $\mathbf{Q} = \mathtt{ADD}(\mathbf{P},\mathbf{Q})$
    $\mathbf{P} = \mathtt{DBL}(\mathbf{P})$
```

Point addition/doubling operations

sequence of finite field operations

DBL: $v_1 = z_1^2, v_2 = x_1 - v_1, \dots$

ADD: $w_1 = z_1^2, w_2 = z_1 \times w_1, \dots$

GF(p) or GF(2^m) operations

operation modulo large prime (GF(p))
or irreducible polynomial (GF(2^m))

Metric for algorithms efficiency: number of multiplications (M) and squares (S) in $\mathbb{F}_p$

ECC versus HECC

	size of $\mathbb{F}_p$ & scalar	ADD	DBL	source
ECC	ℓ_{ECC}	$12\mathtt{M} + 2\mathtt{S}$	$7\mathtt{M} + 3\mathtt{S}$	[1]
HECC	$\ell_{\mathrm{HECC}} \approx \frac{1}{2}\ell_{\mathrm{ECC}}$	$40\mathtt{M} + 4\mathtt{S}$	$38\mathtt{M} + 6\mathtt{S}$	[3]

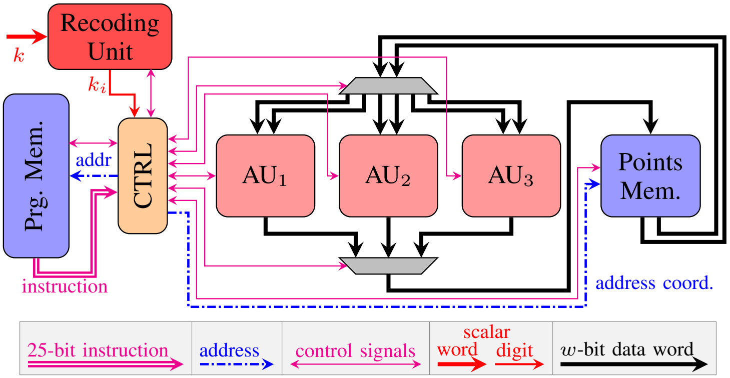
- ECC:
  - Size of $\mathbb{F}_p$ and scalar $2\times$ larger
  - Simpler `ADD` and `DBL` operations
- HECC:
  - Smaller $\mathbb{F}_p$ and scalar
  - More operations in $\mathbb{F}_p$ for `ADD` and `DBL`
- In theory, HECC naturally better than ECC

Objectives

- Design hardware crypto-processors based on customisable architecture
- Implementation of small circuits on FPGA and ASIC
- Study various trade-offs between:
  - Computation speed
  - Area cost
  - Energy consumption and protection against physical attacks
- Experimental comparisons of different architecture configurations
  $\rightarrow$ choice of the architecture parameters for the crypto-processor

Summary

Developed Processor



- Customizable number of arithmetic units over $\mathbb{F}_p$: $\pm$, $\times$, $\div$
  $\rightarrow$ n_M multipliers of size n_B
- w : size of data words

Architecture Parameters

- $w = 32$ bits, for small processors
- 1 adder/subtracter ($\pm$), small and fast
- 1 inverter ($\div$), sufficient for the computations
- n_{M} multipliers ($\times$):
  - Based on Montgomery algorithm for modular multiplication [5]
  - n_{B} : number of parallel active words in the multiplier
  - 3-stage pipeline
- Classical key recoding techniques from literature:
  $\rightarrow$ standard binary, window λNAF methods with $\lambda \in \{2 \ldots 4\}$

Summary

Details of the Experiments

- Objective: compare versions of the processor for various ($n_{\mathtt{M}}$, $n_{\mathtt{B}}$)
- Implementation on Xilinx Spartan 6 LX75 FPGA
- No DSP blocks (for ASIC compatibility)
- Design tools:
  - VHDL and assembly code generation: Python scripts
  - Design implementation: Xilinx design environment ISE 14.6
  - Simulations of complete scalar multiplications with Modelsim
    Theoretical validation with SAGE (with more than 10k vectors)
- Implementation: translate, map, place and route of full processor
- Same optimisation efforts for ECC and HECC

Impact of Key Recoding Unit

- Various recoding techniques proposed to reduce the number of curve operations:
  - BIN: standard binary from left to right
  - NAF: non-adjacent form
  - λNAF: window methods with $\lambda \in \{3, 4\}$
- Implementation results for an ECC processor ($n_{\mathrm{M}} = 1$, $n_{\mathrm{B}} = 1$)

recoding	BIN	NAF	3NAF	4NAF
area, slices (FF/LUT)	517 (1347/1433)	536 (1366/1445)	560 (1370/1454)	547 (1374/1460)
frequency, MHz	229	234	235	231

Results for ECC (256 bits) and HECC (128 bits) (1/2)

	n_M	BRAM	$n_B = 1$		$n_B = 2$		$n_B = 4$	
			area slices (FF/LUT)	freq. MHz	area slices (FF/LUT)	freq. MHz	area slices (FF/LUT)	freq. MHz
ECC	1	3	547 (1374/1460)	231	573 (1476/1625)	233	673 (1674/1875)	233
	2	3	722 (1776/1903)	220	811 (1979/2210)	227	942 (2377/2701)	220
	3	3	810 (2174/2236)	221	915 (2480/2698)	215	1130 (3077/3430)	214
	4	3	952 (2569/2656)	215	1100 (2977/3282)	217	1512 (3771/4293)	216
	5	3	1064 (2982/3136)	210	1405 (3492/3902)	206	1722 (4487/5122)	209
HECC	1	4	514 (1336/1374)	235	549 (1434/1513)	234		
	2	4	646 (1716/1783)	220	737 (1912/2055)	234		
	3	4	732 (2092/2075)	224	826 (2386/2485)	225		
	4	4	870 (2476/2424)	218	1022 (2868/2987)	214		
	5	4	976 (2865/2773)	219	1115 (3355/3465)	210		
	6	4	1089 (3233/3092)	203	1240 (3821/3908)	208		
	7	4	1145 (3601/3426)	213	1372 (4287/4365)	205		
	8	4	1281 (3981/3809)	191	1552 (4765/4890)	183		
	9	4	1379 (4363/4051)	202	1691 (5245/5277)	199		
	10	4	1543 (4739/4435)	196	1856 (5719/5801)	198		
	11	4	1547 (5114/4750)	189	1936 (6192/6240)	198		
	12	4	1738 (5499/5128)	191	2100 (6675/6771)	188		

Results for ECC (256 bits) and HECC (128 bits) (2/2)

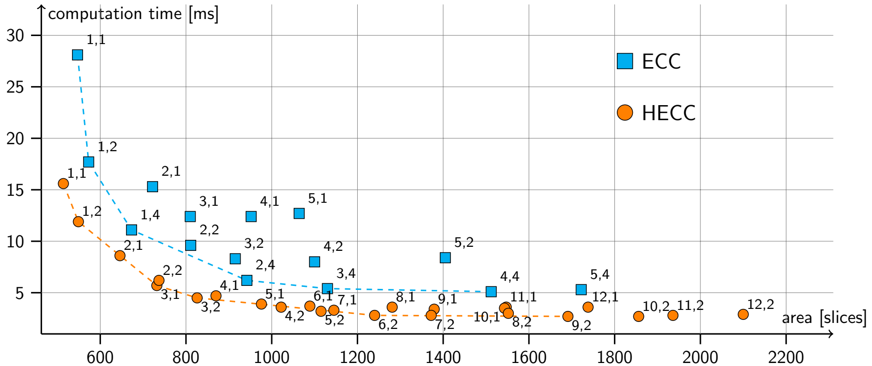
- Average computation time (ms) for 50 $[k]\mathbf{P}$:

	n_B	n_M											
		1	2	3	4	5	6	7	8	9	10	11	12
HECC	1	15.6	8.6	5.7	4.7	3.9	3.7	3.3	3.6	3.4	3.5	3.6	3.6
	2	11.9	6.2	4.5	3.6	3.2	2.8	2.8	3.0	2.7	2.7	2.8	2.9
ECC	1	28.1	15.3	12.4	12.4	12.7							
	2	17.7	9.6	8.3	8.0	8.4							
	4	11.1	6.2	5.4	5.1	5.3							

- Standard deviation for 1000 $[k]\mathbf{P}$:

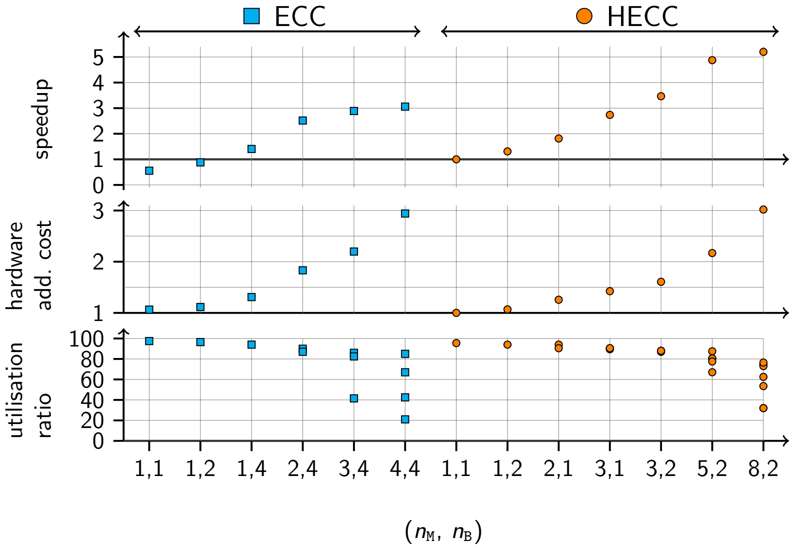
configuration	ECC (1,1)	ECC (3,4)	HECC (1,1)	HECC (6,2)
average time [ms]	28.2	5.3	15.5	2.8
std. deviation [ms]	0.289	0.056	0.324	0.045

Comparison of Various Configurations for our Processor



$(n_{\mathtt{M}},\ n_{\mathtt{B}})$

Relative Efficiency



Comparison with Literature

Source	FPGA	area slices / DSP	freq. MHz	$[k]\mathbf{P}$ duration ms
ECC 1,2	Spartan 6	573 / 0	233	17.7
ECC 1,4		673 / 0	233	11.1
ECC 2,4		942 / 0	220	6.2
ECC 3,4		1 130 / 0	214	5.4
[4]	Virtex-5	1 725 / 37	291	0.38
	Virtex-4	4 655 / 37	250	0.44
[2]	Virtex-4	13 661 / 0	43	9.2
		20 123 / 0	43	7.7

Summary

Conclusion

- Implementation of ECC/HECC processors $\Rightarrow$ cost/performance trade-offs
- We are able to select the appropriate number of multipliers and their size
- Experimental results: HECC always has better performance compared to ECC

Perspectives

- Improve arithmetic units (especially add DSP blocks)
- Study resistance to physical attacks (SCA[1], faults injection) of processors with good time/area trade-offs

[1]Side Channel Attacks

This work is partly funded by
PAVOIS project (ANR-12-BS02-002-01) http://pavois.irisa.fr/
HAH project http://h-a-h.inria.fr/

Thank you for your attention

References

[1] D. J. Bernstein and T. Lange.
Explicit-formulas database.
`http://hyperelliptic.org/EFD/`.

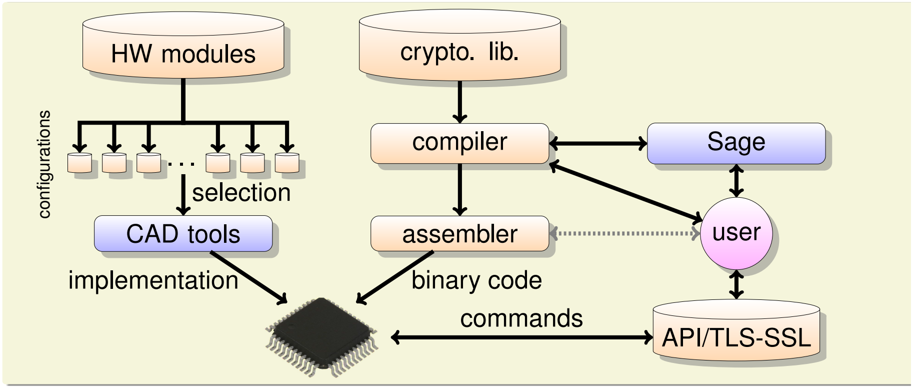
[2] S. Ghosh, M. Alam, D. Roychowdhury, and I.S. Gupta.
Parallel crypto-devices for GF(p) elliptic curve multiplication resistant against side channel attacks.
Computers and Electrical Engineering, 35(2):329–338, March 2009.

[3] T. Lange.
Formulae for Arithmetic on Genus 2 Hyperelliptic Curves.
Applicable Algebra in Engineering, Communication and Computing, 15(5):295–328, February 2005.

[4] Y. Ma, Z. Liu, W. Pan, and J. Jing.
A high-speed elliptic curve cryptographic processor for generic curves over GF(p).
In *Proc. 20th International Workshop on Selected Areas in Cryptography (SAC)*, volume 8282 of *LNCS*, pages 421–437. Springer, August 2013.

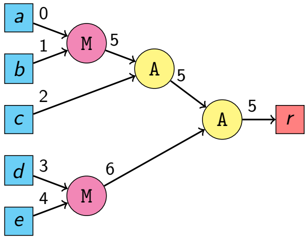
[5] P. L. Montgomery.
Modular multiplication without trial division.
Mathematics of Computation, 44(170):519–521, April 1985.

Implementation Flow



Assembly Language: a Basic Example

$$\boxed{r = (((a \times b) + c) + (d \times e))}$$



```
read fu_mul_0, 0, 1         read operands a and b
launch fu_mul_0             launch ab
read fu_mul_1, 3, 4         read operands d and e
launch fu_mul_1             launch de
wait fu_mul_0               wait until the end of ab
write fu_mul_0, 5           write the result of ab
set OPMODE, 0               set addition (+) mode
read fu_add_sub_0, 5, 2     read ab and operand c
launch fu_add_sub_0         launch (ab) + c
wait fu_mul_1               wait until the end of de
write fu_mul_1, 6           write the result of de
wait fu_add_sub_0           wait until the end of (ab) + c
write fu_add_sub_0, 5       write the result of (ab) + c
read fu_add_sub_0, 5, 6     read (ab) + c and de
launch fu_add_sub_0         launch ((ab) + c) + (de)
wait fu_add_sub_0           wait until the end of ((ab) + c) + (de)
write fu_add_sub_0, 5       write ((ab) + c) + (de)
```

FPGA characteristics

FPGA	Spartan 6	Virtex-4 LX200 [2]	Virtex-5 LX110T [4]	Virtex-4 LX100 [4]
number of slices	11662	89088	17280	49152
number of FF	93296	178176	69120	98304
number of LUT	46648	178176	69120	98304