Hardware and Arithmetic for Hyperelliptic Curves Cryptography

HAH Project

https://h-a-h.cominlabs.u-bretagneloire.fr/

Eval. CominLabs, 2018.05.17–18

HAH Members

Lab-STICC/IRISA:

- Gabriel Gallin*
  PhD student funded by Labex CominLabs
- Audrey Lucas
  PhD student
- Arnaud Tisserand*
  CNRS Researcher

IRMAR:

- Turku Ozlum Celik
  PhD student funded by Labex Lebesgue
- Sylvain Duquesne
  Professor Univ. Rennes 1
- Christophe Ritzenhaler
  Professor Univ. Rennes 1

* with IRISA Lannion $<$ 12-2016 and with Lab-STICC Lorient $\geq$ 12-2016

Introduction

Design of hardware circuits for efficient and secure primitives in public key cryptography (PKC) (or asymmetric crypto.) for small devices

PKC is mandatory to ensure security and privacy in embedded devices, communications, e-commerce, access control, cloud computing, smart-phones, WSN, body area networks, TV boxes, IoT, etc.

PKC primitives:

- key exchange (*e.g.* session keys in symmetric ciphers)
- digital signature (*e.g.* authentication of people and devices)
- specific encryption schemes

Efficiency and security constraints:

- speed, circuit area, energy consumption (max. power)
- robustness against theoretical, logical, and physical attacks

Asymmetric Crypto Solutions

RSA requires very large keys ($\geq$ 2048 bits) is too costly for small devices

Elliptic curves cryptography (ECC) is more efficient than RSA for a similar theoretical security level (*e.g.* 2048b RSA security $\approx$ 220b ECC)

Hyperelliptic curve cryptography (HECC):

- more efficient than ECC (smaller finite fields)
- nice theoretical results
- efficient software implementations (including on small processors)
- but very few secure hardware implementations

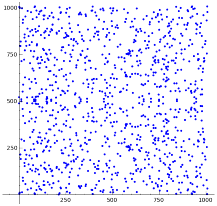
Efficient arithmetic is key for hardware HECC implementations:

- representation(s) of finite field elements
- algorithms for modular arithmetic (*e.g.* $A \times B \bmod P$)
- impact on robustness against physical attacks

HAH Main Objectives

- implement in FPGAs recent results for HECC based on Kummer surfaces
- study and prototype efficient and secure arithmetic algorithms and architectures for HECC on FPGAs
  - arithmetic units (HTMM for modular multiplication)
  - several HECC cryptoprocessors
  - protections against observation attacks
  - trade-offs evaluation between and security
- distribution as open source hardware/software code

Background on (Hyper)-Elliptic Curve Crypto.



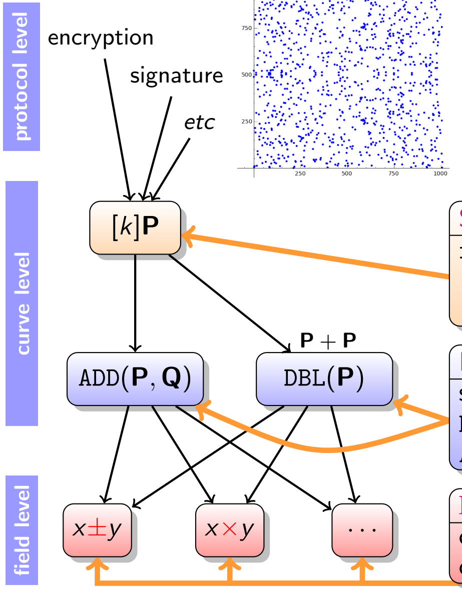
$E : y^2 = x^3 + 4x + 20$ over GF(1009)

points: $\mathbf{P}$, $\mathbf{Q} = (x, y)$ or (x, y, z) or ...

coordinates: $x, y, z \in \mathrm{F}(\cdot)$

$\mathbb{F}_{\mathcal{P}}$, $\mathbb{F}_{2^m}$, t : 200–600 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: $w_1 = z_1^2, w_2 = z_1 \times w_1, \dots$

$\mathbb{F}_{\mathcal{P}}$ or $\mathbb{F}_{2^m}$ operations

operation modulo large prime ($\mathbb{F}_{\mathcal{P}}$)

or irreducible polynomial ($\mathbb{F}_{2^m}$)

ECC, HECC, Kummer-HECC

	$\mathbb{F}_{\mathcal{P}}$ elements size	ADD	DBL	source
ECC	ℓ_{ECC}	$12\mathtt{M} + 2\mathtt{S}$	$7\mathtt{M} + 3\mathtt{S}$	[1]
HECC	$\ell_{\text{HECC}} \approx \frac{1}{2}\ell_{\text{ECC}}$	$40\mathtt{M} + 4\mathtt{S}$	$38\mathtt{M} + 6\mathtt{S}$	[2]
Kummer	ℓ_{HECC}	$19\mathtt{M} + 12\mathtt{S}$		[4]

- ECC:
  - Size of $\mathbb{F}_{\mathcal{P}}$ elements $2\times$ larger
  - Simpler `ADD` and `DBL` operations
- HECC:
  - Smaller $\mathbb{F}_{\mathcal{P}}$
  - More operations in $\mathbb{F}_{\mathcal{P}}$ for `ADD` / `DBL`
- Kummer-HECC is more efficient than ECC [4]:
  - ARM Cortex M0: up to 75% clock cycles reduction for signatures
  - AVR AT-mega: up to 32% cycles reduction for Diffie-Hellman

`M` multiplication, `S` square on field $\mathbb{F}_{\mathcal{P}}$

Scalar Multiplication

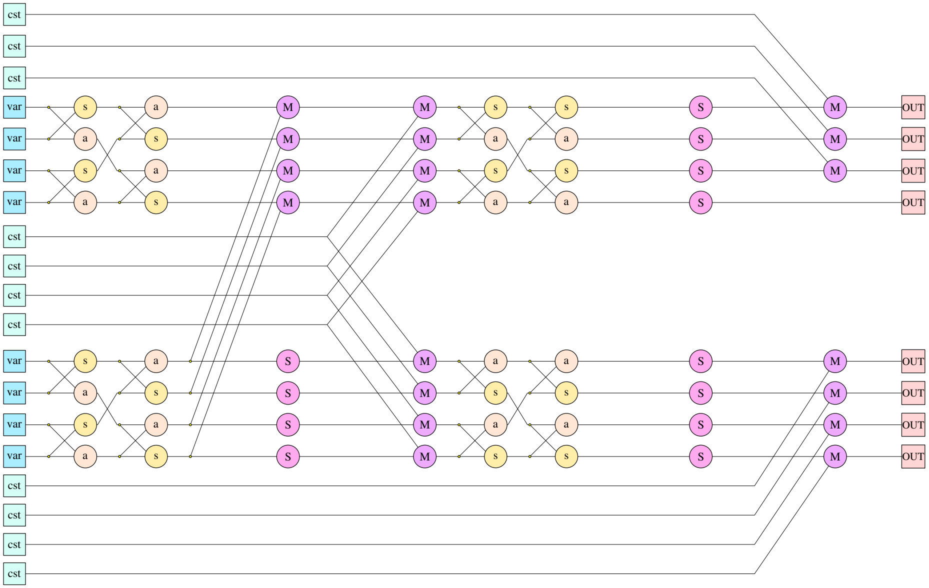
Montgomery ladder based *crypto_scalarmult* [4]:

```
**Require:** $m$-bit scalar $k = \sum_{i=0}^{m-1} 2^i k_i$, point $P_b$, $cst \in \mathbb{F}_{\mathcal{P}}^4$
**Ensure:** $V_1 = [k]P_b$, $V_2 = [k+1]P_b$
  $V_1 \leftarrow cst$
  $V_2 \leftarrow P_b$
  **for** $i = m - 1$ **downto** 0 **do**
    $(V_1, V_2) \leftarrow \texttt{CSWAP}(k_i, (V_1, V_2))$
    $(V_1, V_2) \leftarrow \texttt{xDBLADD}(V_1, V_2, P_b)$
    $(V_1, V_2) \leftarrow \texttt{CSWAP}(k_i, (V_1, V_2))$
  **end for**
  **return** $(V_1, V_2)$
```

$\texttt{CSWAP}(k_i, (X, Y))$ returns (X, Y) if $k_i = 0$, else (Y, X)

- Constant time, uniform operations (independent from key bits)
- Some parallelism between `xDBLADD` internal $\mathbb{F}_{\mathcal{P}}$ operations
- `CSWAP`: very simple but involves secret bits (to be protected)
- Weakness in the original algorithm, we modified it

xDBLADD $\mathbb{F}_{\mathcal{P}}$ Operations



Hyper-Threaded Modular Multiplier (HTMM)

Design of new $\mathbb{F}_{\mathcal{P}}$ multipliers ($A \times B \bmod P$) for performing multiple independent multiplications in the same time (using an hyper-threaded architecture)

First version published at IEEE Asilomar Conference 2017:
15 % speed increase and 45 % area reduction

New version submitted to IEEE Transactions on Computers: 50 % speed increase and 60 % area reduction

Open-source VHDL generator: `https://sourcesup.renater.fr/htmm/`

Architectures Exploration for (H)ECC

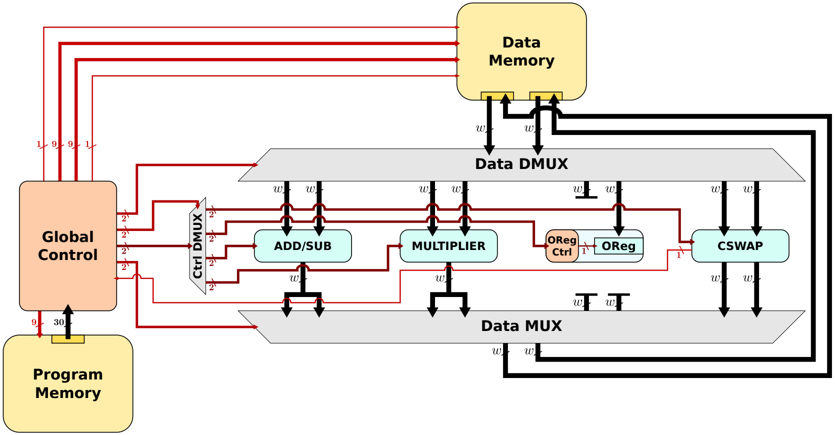
HECC architectures require different types of units:

- $\mathbb{F}_{\mathcal{P}}$ arithmetic units: add/sub, mul, sqr, inv, ...
- Memories, (secure) registers, ...
- Interconnect, global input/output, ...
- Dedicated (secure) control

Problems

- Coding a complete accelerator fully in HDL is costly
- Large design space for various architectures types and parameters (nb. units, algorithms, internal communications and control)
- Need for evaluation of various architectures and parameters
- Need for numerical validation and debug

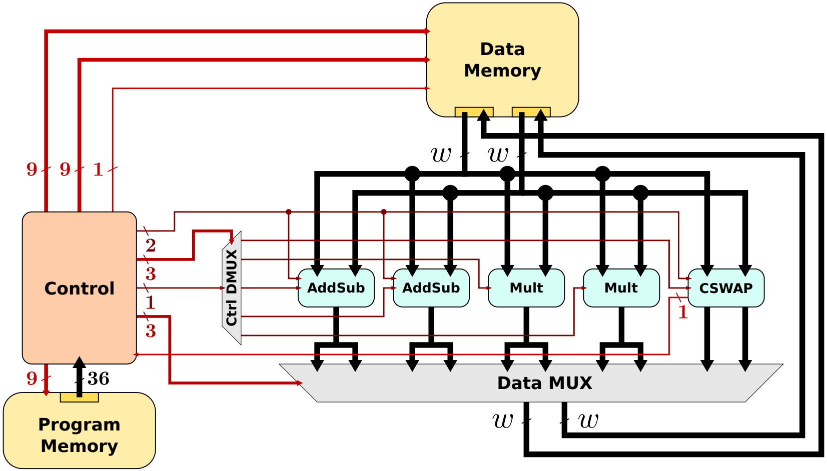
Typical Architecture Model



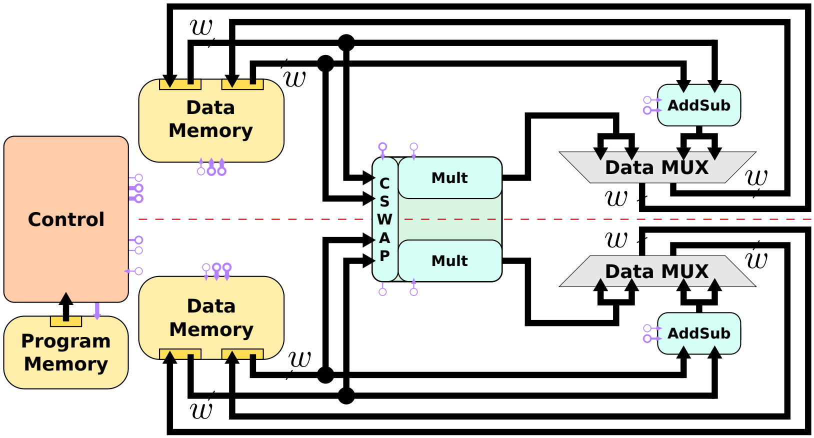
Parameters specified at design time:

- Width w and nb. words s for internal communications ($s \times w = n$)
- Types and number of units

Architecture with more Arithmetic Operators

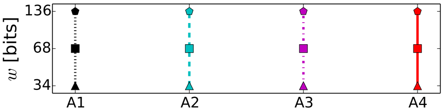


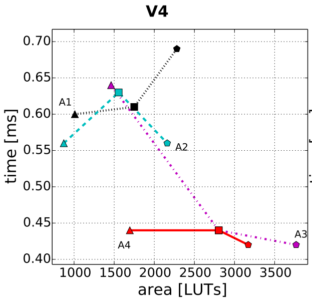
Clustered Architecture

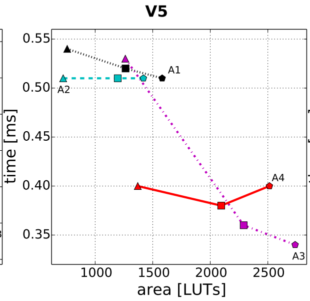


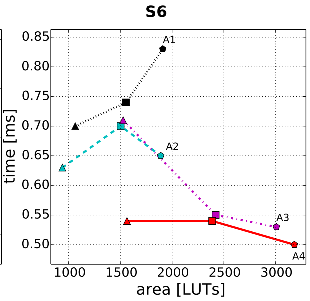
HECC Architectures Comparison (1/2)

Implementations on 3 different FPGAs: Spartan-6 (S6), Virtex-4 (V4) and Virtex-5 (V5).









HECC Architectures Comparison (2/2)

archi.	w [bit]	target	logic slices	DSP blocks	RAM blocks	freq. [MHz]	time [ms]
A2	34	V4	1121	11	4	330	0.56
A3	136		3660	22	9	285	0.42
A4	34		2158	22	7	324	0.44
A2	34	V5	541	11	4	360	0.51
A3	136		1594	22	9	348	0.34
A4	34		1013	22	7	358	0.40
A2	34	S6	381	11	4	293	0.63
A3	136		1131	22	9	225	0.53
A4	34		758	22	7	262	0.54

256b ECC vs 128b HECC (similar theoretical security)

FPGA	Version	DSP	BRAM 18K	Slices	Freq. (MHz)	Nb. cycles	Time (ms)
V4	ECC	37	11	4655	250	109,297	0.44
	HECC_1u	11	7	1413	330	183,051	0.55
	HECC_2u	22	9	2356	330	115,211	0.35
V5	ECC	37	10	1725	291	109,297	0.38
	HECC_1u	11	7	873	360	183,051	0.51
	HECC_2u	22	9	1542	360	115,211	0.32

Gain 1u on V5: -70% DSPs, -30% BRAMs, -49% slices, +30% duration

Gain 2u on V5: -40% DSPs, -10% BRAMs, -10% slices, -15% duration

ECC results from [3]

Conclusion and Perspectives

- HECC is an efficient alternative to ECC for embedded systems
  - More complex formulas but larger internal parallelism
  - Large exploration space for architectures and arithmetic
- Main results
  - Very efficient $\mathbb{F}_{\mathcal{P}}$ multipliers (HTMM), VHDL generator
  - Tools for design space exploration
  - Various architectures to fit design/application constraints
  - Protections against several observation attacks
  - Open source hardware/software code
- Future works
  - Protections against fault injection attacks
  - Explore new architectures

References I

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

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

Yuan Ma, Zongbin Liu, Wuqiong Pan, and Jiwu 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.

Joost Renes, Peter Schwabe, Benjamin Smith, and Lejla Batina.
μKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers.
In *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 9813 of *LNCS*, pages 301–320. Springer, August 2016.

Proposed Design Framework

- Hierarchical description and simulation for HECC architectures at CCABA level (Critical-Cycle Accurate, Bit Accurate)
  - Units inputs/outputs are bit accurate
  - Units inputs/outputs and external control are critical cycle accurate
- Description of various architectures at high-level
  - Composition of units for different parameters and optimizations
  - Scheduling tool for control and communications (*work in progress*)
- Units described, optimized and validated in HDL
  - Perfectly known behavior $\rightarrow$ no need for cycle accurate simulation
  - Area, latency, . . . come from actual FPGA implementation
- Dedicated simulator in Python
  - Fast development and numerical validation
  - Sage (`http://www.sagemath.org/`) interface for HECC support

Configuration for Implementations

- 128 bits HECC solutions
- $\mathbb{F}_{\mathcal{P}}$ adder-subtractor (AddSub):
  - 4 cycles latency pipeline
  - $8 \cdots 11$ cycles delay depending on w
- $\mathbb{F}_{\mathcal{P}}$ multiplier (HTMM):
  - Hyper-threaded multiplier for 3 sets of operands computed in parallel
  - 5 cycles latency for loading and reading
  - $68 \cdots 71$ cycles delay depending on w
- CSWAP unit:
  - Secure management of key bits
  - $2 \cdots 4$ cycles delay depending on w

Results for Basic Architecture (1 Add/Sub, 1 HTMM)

Version $s \times w$	Clock cycles	Units	DSP	BRAM	FF	LUT	Slices	RAM #lines
4x34	207,383	HTMM	11	2	587	359	180	12
		AddSub	0	0	366	226	80	-
		DATA_MEM	0	1	0	0	0	112
		PRGM_MEM	0	1	0	0	0	208
		CSWAP	0	0	536	290	103	-
2x68	185,615	HTMM	11	2	970	633	315	12
		AddSub	0	0	713	382	148	-
		DATA_MEM	0	2	0	0	0	56
		PRGM_MEM	0	1	0	0	0	234
		CSWAP	0	0	553	297	122	-
1x136	183,051	HTMM	11	2	1066	623	309	12
		AddSub	0	0	784	464	212	-
		DATA_MEM	0	4	0	0	0	26
		PRGM_MEM	0	1	0	0	0	250
		CSWAP	0	0	685	431	155	-

s: number of words, w: size of words

Increasing the Number of Arithmetic Units

Version $s \times w$	Clock cycles	Units	DSP	BRAM	FF	LUT	Slices	RAM #lines
4x34	**203,543**	HTMM **x 2**	**22**	**4**	**1174**	**718**	**360**	12
		ADDSUB **x 2**	0	0	**732**	**452**	**160**	-
		DATA_MEM	0	1	0	0	0	**108**
		PRGM_MEM	0	1	0	0	0	**213**
		CSWAP	0	0	536	290	103	-
2x68	**125,455**	HTMM **x 2**	**22**	**4**	**1940**	**1266**	**630**	12
		ADDSUB **x 2**	0	0	**1426**	**764**	**296**	-
		DATA_MEM	0	4	0	0	0	**50**
		PRGM_MEM	0	1	0	0	0	**211**
		CSWAP	0	0	553	297	122	-
1x136	**115,211**	HTMM **x 2**	**22**	**4**	**2132**	**1246**	**618**	12
		ADDSUB **x 2**	0	0	**1568**	**928**	**424**	-
		DATA_MEM	0	4	0	0	0	**25**
		PRGM_MEM	0	1	0	0	0	**235**
		CSWAP	0	0	685	431	155	-

s: number of words, w: size of words