ESC 2013, Mondorf-les-Bains, Luxembourg, January 18, 2013

Hardware/Software Co-Design of Lightweight Elliptic Curve Cryptography for the Internet of Things

Johann Großschädl

UNIVERSITÉ DU LUXEMBOURG

Laboratory of Algorithmics, Cryptology and Security (LACS)
University of Luxembourg
http://lacs.uni.lu/members/johann_groszschaedl

Motivation

Vehicle, asset, person & pet controlling and monitoring

Agriculture automation

Energy consumption

Security & surveillance

Building management

M2M & wireless sensor network

Internet of Things
Smart Objects
Wirelessly connected

Embedded mobile

Everyday things

Smart homes

Telemedicine & healthcare

Source: nkonnect.com

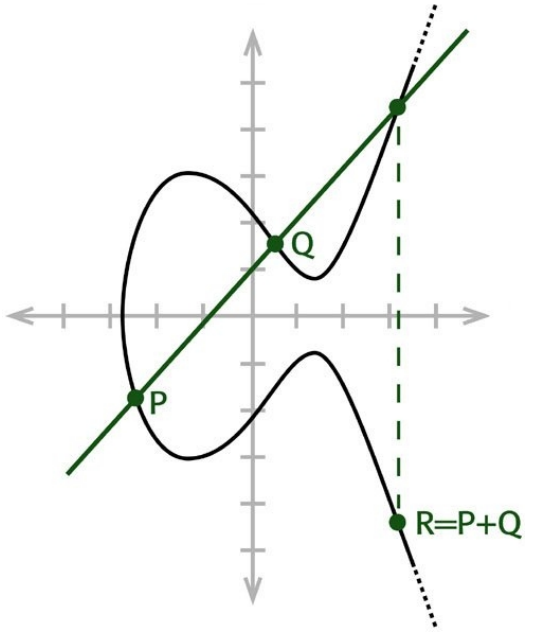
- Processors embedded into everyday objects (“things”)
- Wireless communication (WiFi, Bluetooth, ZigBee)

Outline

- ECC software on ATmega128
  - Optimizations of field and point arithmetic
  - Analysis of execution time
  - Bottlenecks of ATmega128
- New AVR processor: JAAVR
  - ATmega128 clone
  - Enhancements to speed up prime-field arithmetic
  - Execution time and silicon area
- Example: ECDH key exchange
  - Analysis of energy cost

Elliptic Curve Cryptography

- Requirements
  - Performance (fast execution time)
  - Memory, code size, power, etc.
- Group (point) arithmetic
  - Speed vs memory trade-off
  - This work: no pre-computed points
- Field arithmetic
  - Speed vs memory vs code size
  - This work: no loop unrolling
  - Size of RAM and ROM determine silicon area (price)



Special Curves

- Weierstraß Form: $y^2 = x^3 + ax + b$
  - Point-Add: 8M+3S, Point-Dbl: 4M+4S
- Montgomery Form: $by^2 = x^3 + ax^2 + x$
  - Point-Add: 3M+2S, Point-Dbl: 2M+2S
  - Ladder: 1 Point-Add and 1 Point-Dbl per scalar-bit
- Twisted Edwards Form: $ax^2 + y^2 = 1 + dx^2y^2$
  - Point-Add: 7M, Point-Dbl: 3M+4S
- GLV Curves: $y^2 = x^3 + ax$ or $y^2 = x^3 + b$
  - Weierstraß curve with either $a = 0$ or $b = 0$
  - Efficiently computable endomorphism

Prime Fields

- Generalized Mersenne prime fields
  - E.g. $p = 2^{160} - 2^{31} - 1$ = 0xffffffffffffffffffffffffffffffff7fffffff
  - Modular reduction through shifts and additions
- Pseudo-Mersenne prime fields
  - E.g. $p = 2^{160} - 47$ = 0xffffffffffffffffffffffffffffffffffffffd1
  - Reduction through multiplication by a small constant
- Optimal Prime Fields (OPFs)
  - Low-weight prime $p = u \cdot 2^k + 1$ where u is small (8 bits)
  - E.g. $p = 126 \cdot 2^{152} + 1$ = 0x7e00000000···000000001
  - Montgomery reduction is fast for low-weight primes

Multiplication in Prime Fields

- 160-bit operands on 8-bit processor
  - Operands stored in arrays of b = 160/8 = 20 bytes
- Pseudo-Mersenne prime field
  - Multiplication (320-bit product): b^2 = 400 mul instr.
  - Thereafter reduction: b = 20 mul instructions
- Optimal Prime Field (OPF)
  - FIPS method for Montgomery multiplication
  - Multiplication and reduction are interleaved
  - $b^2 + b$ = 420 mul instructions for our low-weight prime
  - Needs less memory and fewer load/store instr.

Implementation on 8-bit AVR

- ATmega128 processor
  - 8-bit RISC, 32 general-purpose registers
  - mul instr.: 2 cycles, load/store instr.: 2–3 cycles
- 160-bit OPF multiplication
  - 3314 clock cycles
  - Hybrid multiplication (4 bytes per iteration of loop)
  - 3000 cycles with loop unrolling and Karatsuba
- Main bottlenecks
  - Multi-cycle latency of certain instructions (e.g. loads)
  - Small (i.e. inefficient) integer multiplier

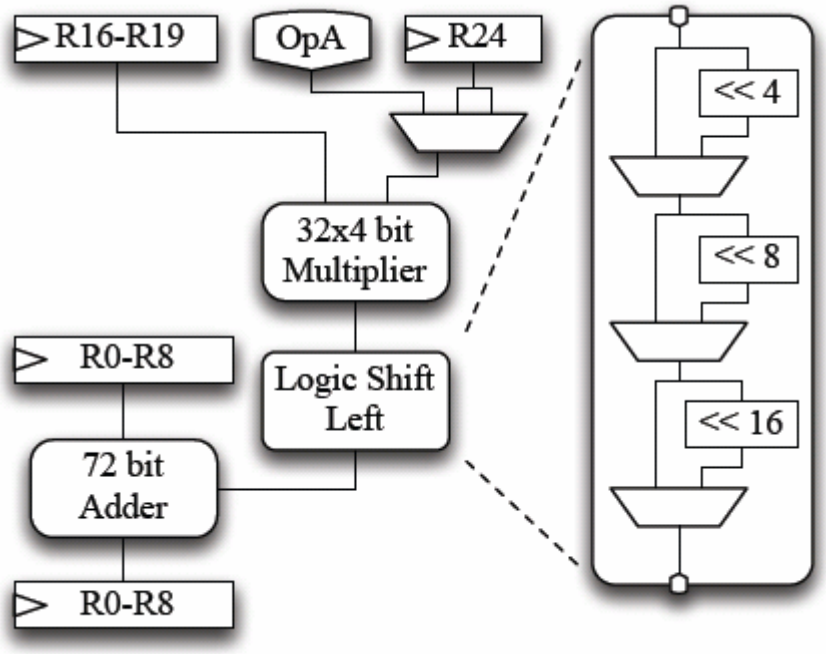
Our AVR Processor: JAAVR

- Just Another AVR
  - ATmega128 clone written in VHDL
  - Supports full AVR instruction set

- Improved Instruction Timing
  - Special mode of operation with 1-cycle latency
  - Can be switched between ATmega128-compatible CPI and improved CPI
- (32x4)-bit Multiply/Accumulate Unit
  - Multiplier operates on general-purpose registers
  - Designed to speed up multi-precision multiplication

Multiply/Accumulate Unit

- Main Components
  - (32x4)-bit multiplier
  - 72-bit accumulator
- Tightly Integrated
  - Operand from GPRs
  - r16-r19 and r24
  - r0 t0 r9 for result



- (32x32)-bit Multiplication
  - 8 clock cycles
  - Load instr. can be executed during a multiplication

OPF Arithmetic on JAAVR

- Inner-Loop operation
  - (32x32+72)-bit MAC operation takes 8 cycles
  - 8 bytes for next MAC loaded during current MAC
  - Two ways of accessing MAC unit (see paper)
  - 80-90 clock cycles on “conventional” ATmega128
- Multiplication in 160-bit OPF
  - Baseline implementation: 3314 cycles
  - 552 cycles in JAAVR when using (32x4)-bit multiplier
- Addition, Subtraction, Inversion
  - Slightly faster on JAAVR due to improved CPI

OPF Timings and Core Area

TABLE I
OPF OPERATIONS ON DIFFERENT VARIANTS OF JAAVR.

Processor Mode	CA	FAST	ISE
Runtimes [Cycles]			
Addition	240	145	145
Subtraction	240	145	145
Multiplication	3,314	2,537	552
Inversion	189k	128k	124k
Chip Area			
JAAVR	6,166 GE	6,800 GE	8,344 GE
Difference	-	+10%	+35%

Scalar Multiplication Timings

TABLE II
POINT MULTIPLICATION TIMES ON A STANDARD ATMEGA128.

Elliptic Curve	High-speed Method	High-speed Runtime [kCycles]	Constant Round Method	Constant Round Runtime [kCycles]
Standardized Curve				
secp160r1	NAF	7,136	Mon	8,722
Non-Standardized Curves using OPF				
Weierstraß	NAF	6,983	Mon	8,824
Edwards	NAF	5,597	DAAA	8,251
Montgomery	Mon	5,545	Mon	**5,545**
GLV	BHL JSF	**3,930**	Mon	8,132

Constant Round: constant execution time (SPA resistant)

Overall Area (incl. RAM+ROM)

Elliptic Curve	Mode	Point Mult. [Cycles]	ROM [bytes]	JAAVR [GE]	ROM [GE]	RAM [GE]	Total [GE]
Weierstraß	CA	6,982,629	6,224	6,166	9,091	4,485	19,742
Edwards	CA	5,596,860	6,022	6,166	8,694	4,712	**19,572**
Montgomery	CA	5,545,078	6,824	6,167	9,542	4,359	20,068
GLV	CA	3,930,256	8,638	6,166	12,413	6,450	25,029
Weierstraß	FAST	5,254,706	6,224	6,800	9,071	4,485	20,355
Edwards	FAST	4,214,289	6,022	6,802	8,695	4,712	20,208
Montgomery	FAST	4,165,405	6,824	6,803	9,533	4,359	20,695
GLV	FAST	2,939,929	8,638	6,802	12,413	6,450	25,665
Weierstraß	ISE	1,542,981	6,290	8,344	8,718	4,485	21,546
Edwards	ISE	1,230,663	6,128	8,345	8,562	4,359	21,266
Montgomery	ISE	1,299,598	**5,752**	8,343	7,926	4,712	**20,980**
GLV	ISE	**1,001,302**	8,640	8,330	12,078	6,450	26,858

- GLV curve is fastest when SPA resistance is not needed
- Montgomery curve is fairly fast and has no SPA leakage
- Twisted Edwards curve has best area-time product

Energy Evaluation

- Scalar multiplication on JAAVR
  - Power @ 7.4 MHz: 125 – 163 µW (processor core), 8.9 – 40 µW (RAM), roughly 80 µW (ROM)
  - Max. overall power: 283 µW
  - Execution time @ 7.4 MHz: 135 ms (GLV curve)
  - Energy cost of 1 scalar multiplication: 38.2 µJ
- ECDH key exchange between MICAz motes
  - 2 scalar multiplications: 76.4 µJ
  - Energy of two 1.5V AA batteries: 21,600 J
  - $280 \cdot 10^6$ key exchanges before running out of battery

Conclusions

- ECC on JAAVR
  - 4 different families of elliptic curve
  - Arithmetic in 160-bit OPF
  - Hybrid method for multi-precision multiplication
- Enhancements of JAAVR
  - Improved CPI and (32x4+72)-bit MAC unit
  - Increase core area by only 2178 gates
  - Improve scalar multiplication by factor of 4
- Which curve to use?
  - No clear winner, depends on requirements