Arithmetic Operators for Pairing-Based Cryptography

Jean-Luc Beuchat

Laboratory of Cryptography and Information Security
Graduate School of Systems and Information Engineering
University of Tsukuba
1-1-1 Tennodai, Tsukuba
Ibaraki, 305-8573, Japan
mailto:beuchat@risk.tsukuba.ac.jp

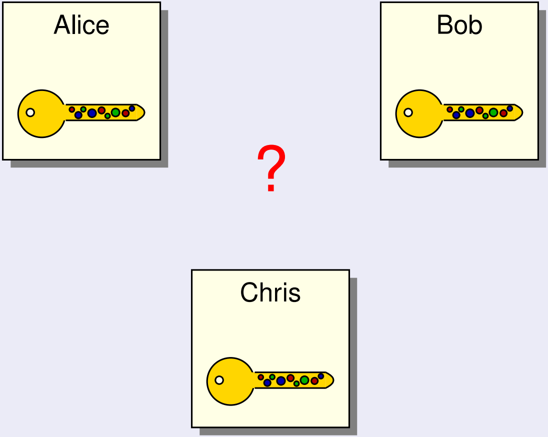
Joint work with Nicolas Brisebarre (Université J. Monnet, Saint-Étienne, France), Jérémie Detrey (ENS Lyon, France), Eiji Okamoto (University of Tsukuba, Japan), Masaaki Shirase (Future University, Hakodate, Japan), and Tsuyoshi Takagi (Future University, Hakodate, Japan)

Outline of the Talk

Example: Three-Party Key Agreement

Key agreement

How can Alice, Bob, and Chris agree upon a shared secret key?



Example: Three-Party Key Agreement

Discrete logarithm problem (DLP)

- $G = \langle P \rangle$: additively-written group of order n
- DLP: given P, Q, find the integer $x \in \{0, \dots, n-1\}$ such that $Q = xP$

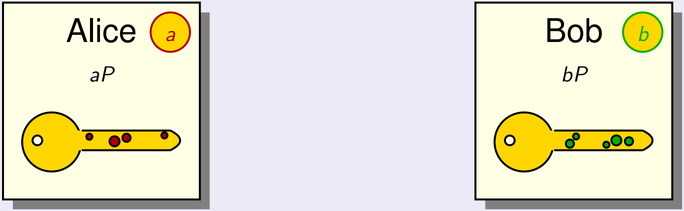
Example: Three-Party Key Agreement

Discrete logarithm problem (DLP)

- $G = \langle P \rangle$: additively-written group of order n
- DLP: given P, Q, find the integer $x \in \{0, \ldots, n-1\}$ such that $Q = xP$

Diffie-Hellman problem (DHP)

Given P, aP, and bP, find abP.



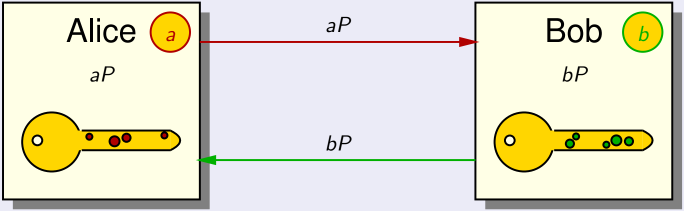
Example: Three-Party Key Agreement

Discrete logarithm problem (DLP)

- $G = \langle P \rangle$: additively-written group of order n
- DLP: given P, Q, find the integer $x \in \{0, \ldots, n-1\}$ such that $Q = xP$

Diffie-Hellman problem (DHP)

Given P, aP, and bP, find abP.



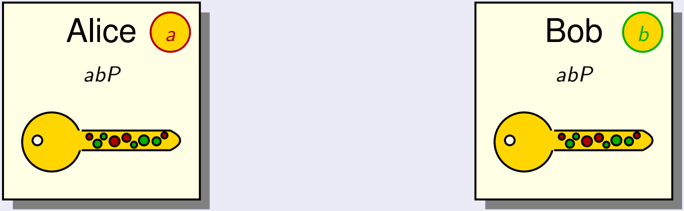
Example: Three-Party Key Agreement

Discrete logarithm problem (DLP)

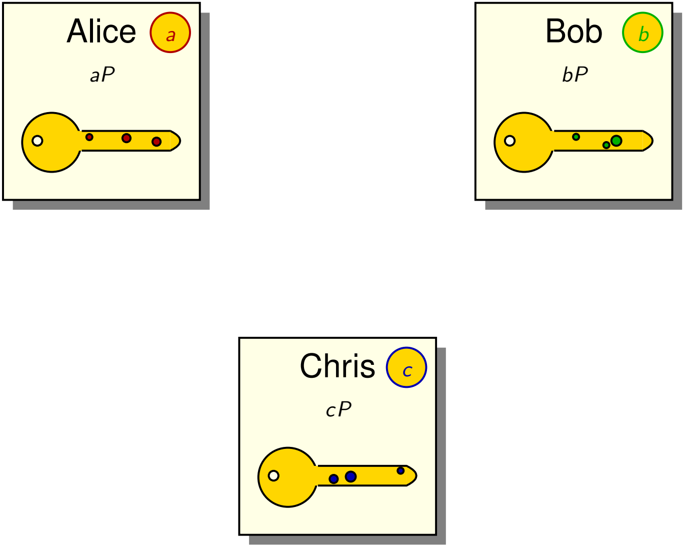
- $G = \langle P \rangle$: additively-written group of order n
- DLP: given P, Q, find the integer $x \in \{0, \ldots, n-1\}$ such that $Q = xP$

Diffie-Hellman problem (DHP)

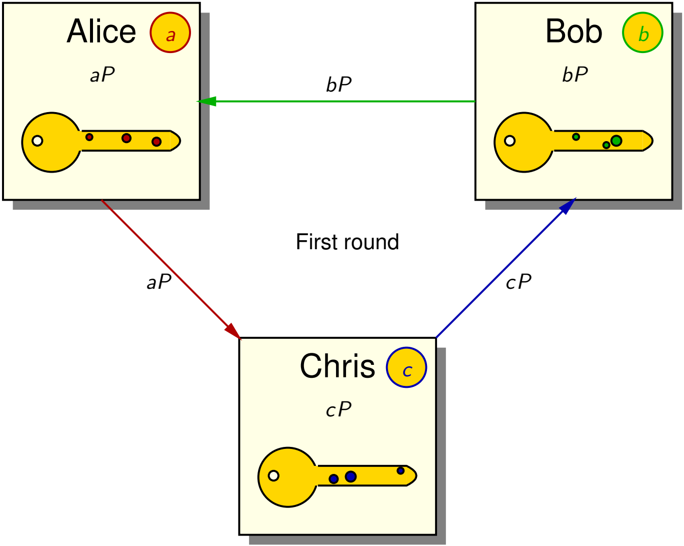
Given P, aP, and bP, find abP.



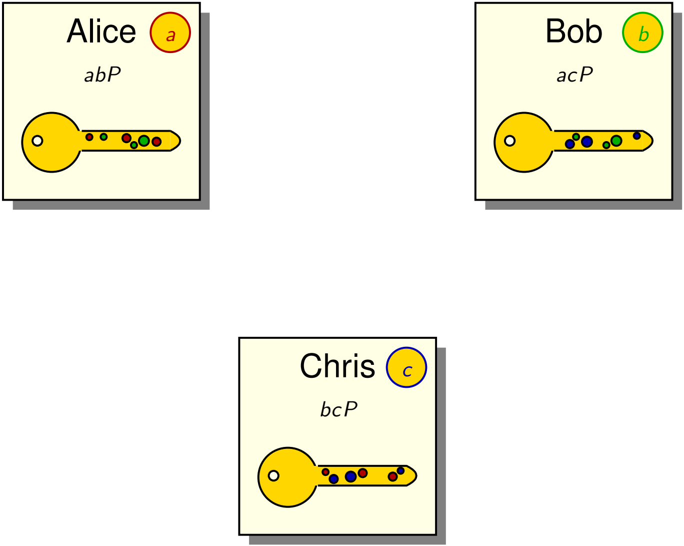
Example: Three-Party Key Agreement



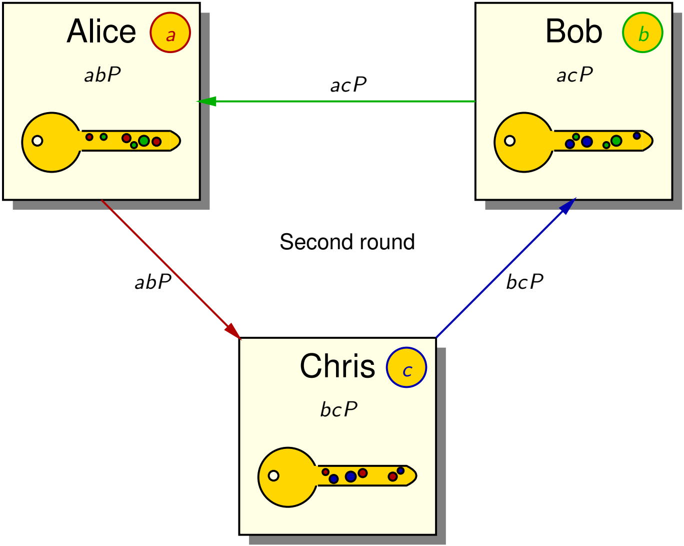
Example: Three-Party Key Agreement



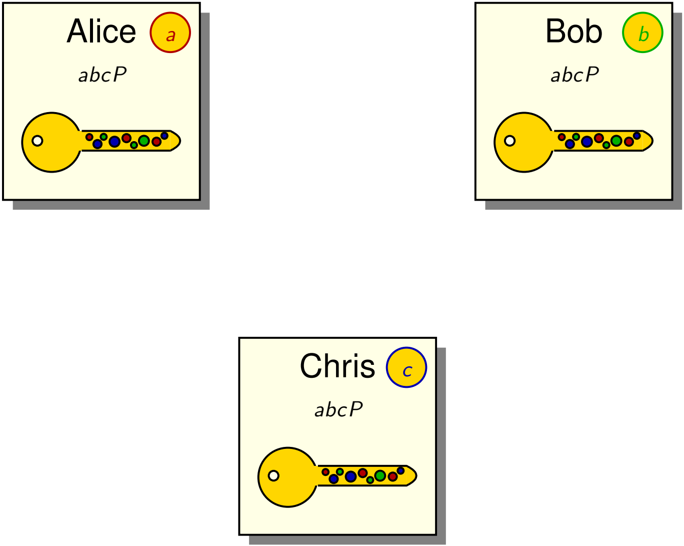
Example: Three-Party Key Agreement



Example: Three-Party Key Agreement



Example: Three-Party Key Agreement



Example: Three-Party Key Agreement

Three-party two-round key agreement protocol

Does a three-party one-round key agreement protocol exist?

Example: Three-Party Key Agreement

Bilinear pairing

- $G_1 = \langle P \rangle$: additively-written group
- G_2: multiplicatively-written group with identity 1
- A bilinear pairing on (G_1, G_2) is a map

$$\hat{e} : G_1 \times G_1 \longrightarrow G_2$$

that satisfies the following conditions:

1. **Bilinearity.** For all Q, R, $S \in G_1$,

$$\hat{e}(Q + R, S) = \hat{e}(Q, S)\hat{e}(R, S) \quad \text{and} \quad \hat{e}(Q, R + S) = \hat{e}(Q, R)\hat{e}(Q, S).$$

2. **Non-degeneracy.** $\hat{e}(P, P) \neq 1$.
3. **Computability.** $\hat{e}$ can be efficiently computed.

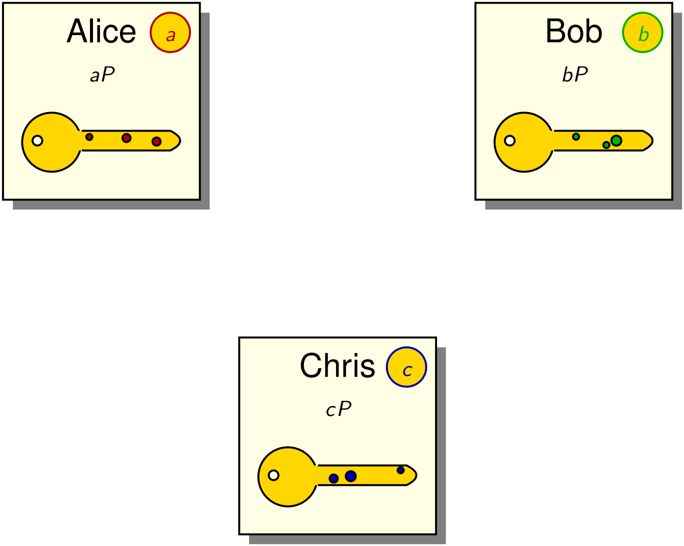
Example: Three-Party Key Agreement

Bilinear Diffie-Hellman problem (BDHP)

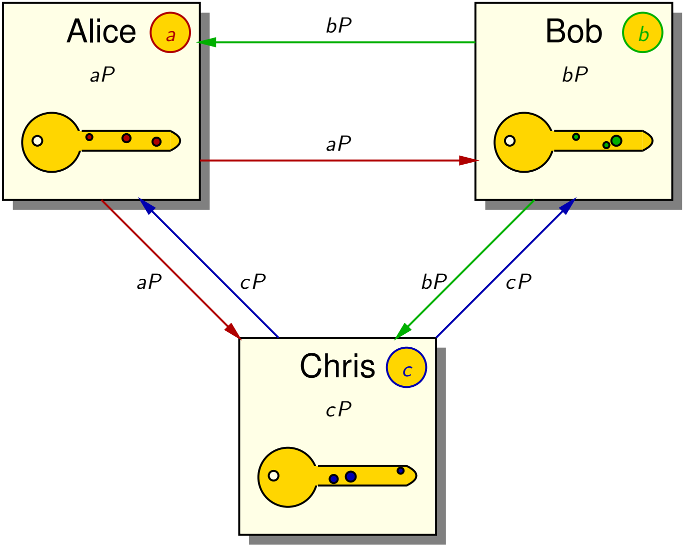
Given P, aP, bP, and cP, compute $\hat{e}(P, P)^{abc}$

Assumption: the BDHP is difficult

Example: Three-Party Key Agreement



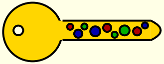
Example: Three-Party Key Agreement



Example: Three-Party Key Agreement

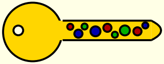
Alice a

$\hat{e}(bP, cP)^a$



Bob b

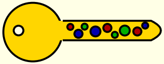
$\hat{e}(aP, cP)^b$



$$\hat{e}(bP, cP)^a = \hat{e}(aP, cP)^b = \hat{e}(aP, bP)^c = \hat{e}(P, P)^{abc}$$

Chris c

$\hat{e}(aP, bP)^c$



Example: Three-Party Key Agreement

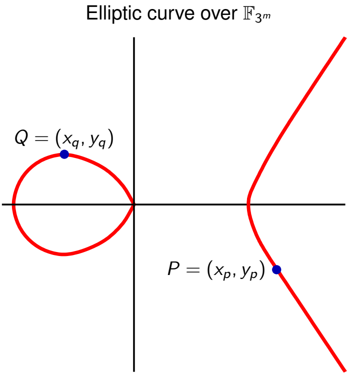
Examples of cryptographic bilinear maps

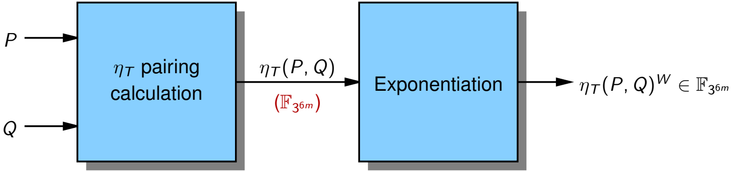
- Weil pairing
- Tate pairing
- η_T pairing (Barreto *et al.*)
- Ate pairing (Hess *et al.*)

Applications

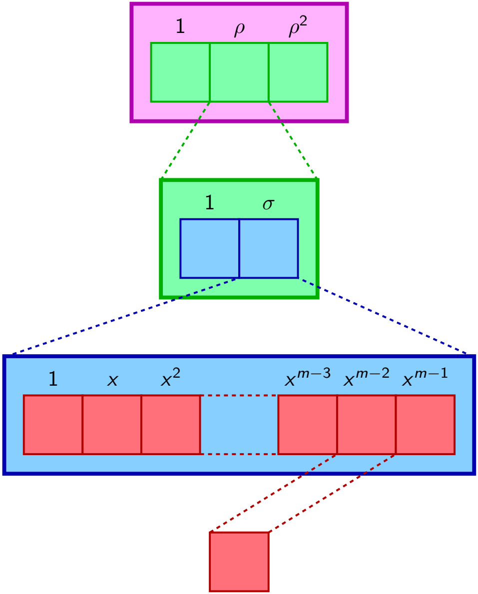
- Identity based encryption
- Short signature

Computation of the η_T Pairing





Computation of the η_T Pairing – Tower Field



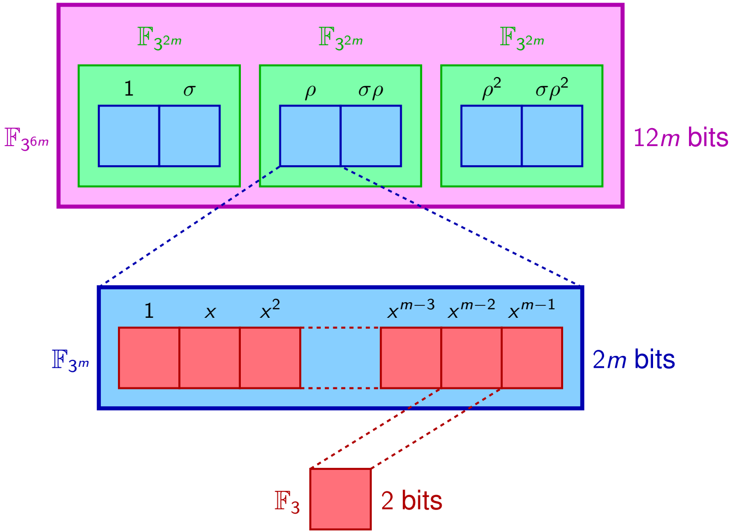
$$\mathbb{F}_{3^{6m}} = \mathbb{F}_{3^{2m}}[\rho]/(\rho^3 - \rho - 1)$$

$$\mathbb{F}_{3^{2m}} = \mathbb{F}_{3^m}[\sigma]/(\sigma^2 + 1)$$

$$\mathbb{F}_{3^m} = \mathbb{F}_3[x]/(f(x))$$

$$\mathbb{F}_3 = \mathbb{Z}/3\mathbb{Z} = \{0, 1, 2\}$$

Computation of the η_T Pairing – Tower Field



Computation of the η_T Pairing

$\eta_T(P, Q)$

- Addition
- Multiplication
- Cubing
- Cube root

$\eta_T(P, Q)^{3^{\frac{m+1}{2}}}$ (Arith 18)

- Addition
- Multiplication
- Cubing

Bilinearity of $\eta_T(P, Q)^W$

$$\eta_T\left(P, Q\right)^W = \sqrt[3^m]{\left(\eta_T\left(\left[3^{\frac{m-1}{2}}\right] P, Q\right)^{3^{\frac{m+1}{2}}}\right)^W}$$

Computation of the η_T Pairing

Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

- $\frac{m+1}{2}$ multiplications
- Operands: A and $B \in \mathbb{F}_{3^{6m}}$ with

$B =$

1	σ	ρ	$\sigma\rho$	ρ^2	$\sigma\rho^2$
$-r_0^2$	$y_p y_q$	$-r_0$	0	-1	0

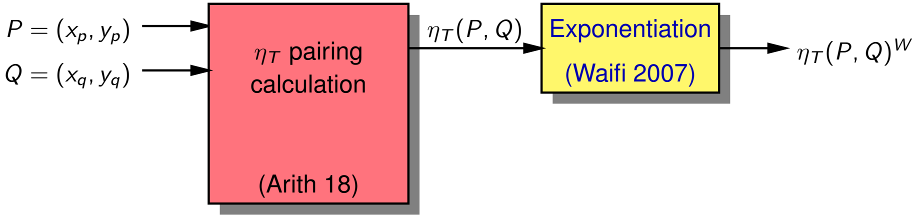
r_0, y_p, and $y_q \in \mathbb{F}_{3^m}$

- Cost: 13 multiplications and 46 additions over $\mathbb{F}_{3^m}$

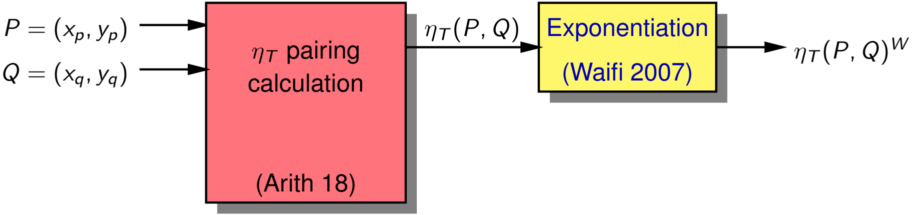
Multiplication over $\mathbb{F}_{3^{6m}}$ – Exponentiation

- Only one multiplication
- Operands: A and $B \in \mathbb{F}_{3^{6m}}$
- Cost: 18 multiplications and 58 additions over $\mathbb{F}_{3^m}$

A Coprocessor for the η_T Pairing Computation



A Coprocessor for the η_T Pairing Computation



Computation of $\eta_T(P, Q)$: multiplication over $\mathbb{F}_{3^{6m}}$

- New algorithm
  - 15 multiplications and 29 additions over $\mathbb{F}_{3^m}$
  - Allows one to share operands between multipliers (less registers)
- Architecture
  - 9 multipliers
  - Most significant coefficient first (Horner's rule)

A Coprocessor for the η_T Pairing Computation

Prototype

- Field: $\mathbb{F}_{3^{97}} = \mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$
- FPGA: Cyclone II EP2C35 (Altera)

A Coprocessor for the η_T Pairing Computation

Prototype

- Field: $\mathbb{F}_{3^{97}} = \mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$
- FPGA: Cyclone II EP2C35 (Altera)

$\eta_T(P, Q)$ (Arith 18)

- Arithmetic over $\mathbb{F}_{3^{97}}$
  - 9 multipliers
  - 2 adders
  - 1 cubing unit
- Area: 14895 LEs
- Frequency: 149 MHz
- Computation time: 33 μs

A Coprocessor for the η_T Pairing Computation

Prototype

- Field: $\mathbb{F}_{3^{97}} = \mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$
- FPGA: Cyclone II EP2C35 (Altera)

$\eta_T(P, Q)$ (Arith 18)

- Arithmetic over $\mathbb{F}_{3^{97}}$
  - 9 multipliers
  - 2 adders
  - 1 cubing unit
- Area: 14895 LEs
- Frequency: 149 MHz
- Computation time: 33 μs

Exponentiation (Waifi 2007)

Challenge

Raise $\eta_T(P, Q)$ to the W power

- in 33 μs (or less)
- with the smallest amount of hardware

A Coprocessor for the η_T Pairing Computation

Why FPGAs?

- Prototyping

A Coprocessor for the η_T Pairing Computation

Why FPGAs?

- Prototyping
- Short time to market

A Coprocessor for the η_T Pairing Computation

Why FPGAs?

- Prototyping
- Short time to market
- Small series

A Coprocessor for the η_T Pairing Computation

Why FPGAs?

- Prototyping
- Short time to market
- Small series
- Hardware accelerators for some applications (e.g. cryptography)

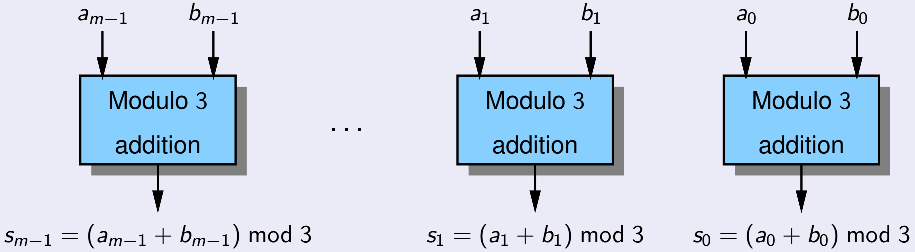
A Coprocessor for the Final Exponentiation

Final exponentiation: operations over $\mathbb{F}_{3^m}$

Additions	477
Multiplications	78
Cubings	$3m + 3$
Inversion	1

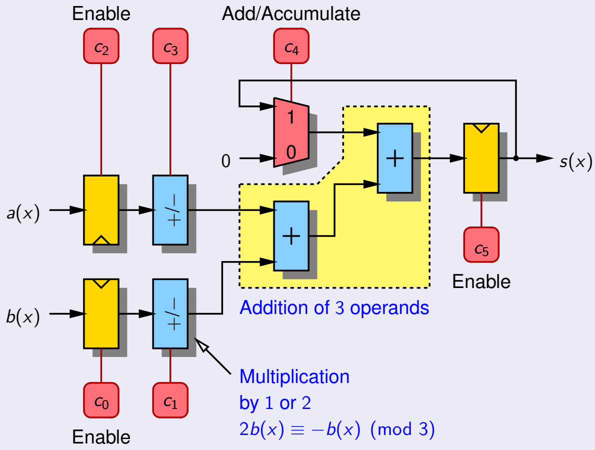
A Coprocessor for the Final Exponentiation

Addition over $\mathbb{F}_{3^m}$



A Coprocessor for the Final Exponentiation

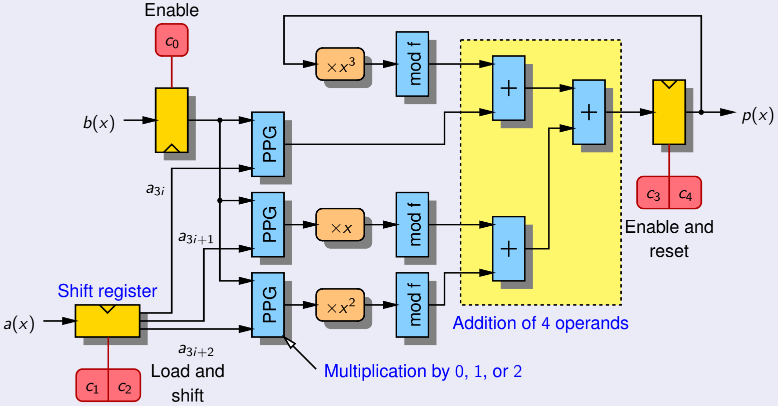
Addition, subtraction, and accumulation over $\mathbb{F}_{3^m}$



A Coprocessor for the Final Exponentiation

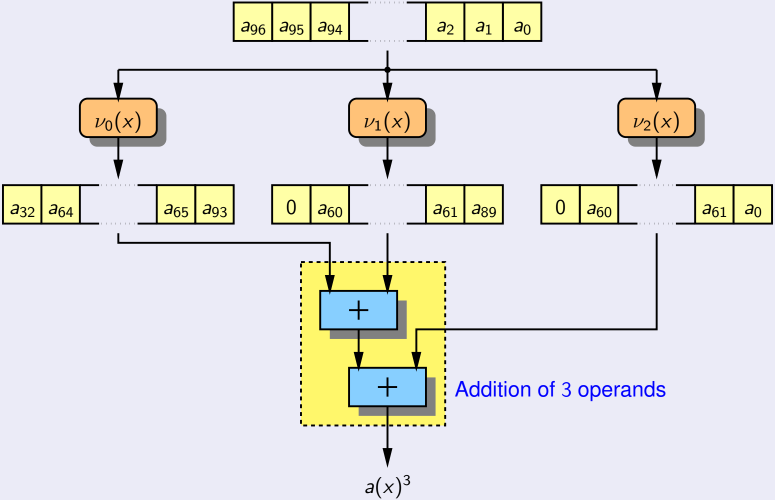
Multiplication over $\mathbb{F}_{3^m}$

- Array multiplier ($\lceil m/3 \rceil$ clock cycles)
- Most significant coefficient first (Horner's rule)



A Coprocessor for the Final Exponentiation

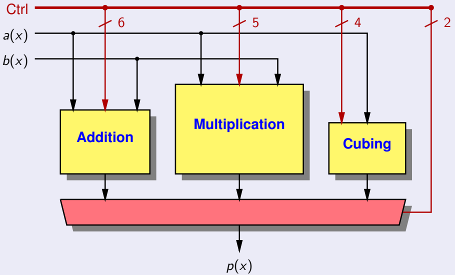
Cubing over $\mathbb{F}_3[x]/(x^{97}+x^{12}+2)$



A Coprocessor for the Final Exponentiation

Arithmetic operators over $\mathbb{F}_{3^{97}}$ on a Cyclone II FPGA

Operation	Area [LEs]	Control [bits]
Add./sub.	970	6
Mult.	1375	5
Cubing	668	4
ALU	3308	17



A Coprocessor for the Final Exponentiation

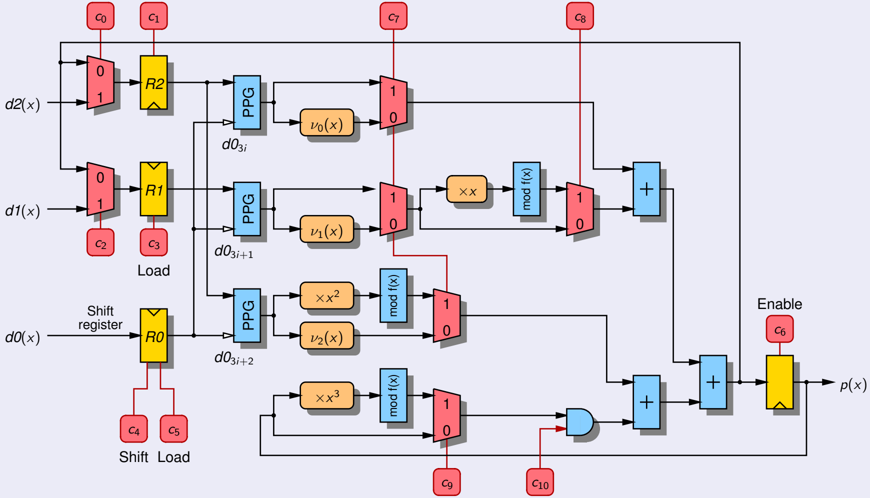
Unified arithmetic operator

- Operations
  - Addition
  - Subtraction
  - Accumulation
  - Multiplication
  - Cubing
- Area (Cyclone II): 2676 LEs (instead of 3308)
- Control bits: 11 (instead of 17)
- Inversion: Fermat's little theorem (96 cubings and 9 multiplications)

$$a^{3^m-2} = a^{-1}, \text{ where } a \in \mathbb{F}_{3^m}$$

A Coprocessor for the Final Exponentiation

Unified arithmetic operator



A Coprocessor for the Final Exponentiation

Prototype

- Field: $\mathbb{F}_{3^{97}} = \mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$
- FPGA: Cyclone II EP2C35 (Altera)

$\eta_T(P, Q)$ (Arith 18)

- Arithmetic over $\mathbb{F}_{3^{97}}$
  - 9 multipliers
  - 2 adders
  - 1 cubing unit
- Area: 14895 LEs
- Frequency: 149 MHz
- Computation time: 33 μs

Exponentiation (Waifi 2007)

- Unified operator
- Area: 2787 LEs
- Frequency: 159 MHz
- Computation time: 26 μs

A Coprocessor for the Full Pairing Computation

Operations over $\mathbb{F}_{3^m}$

Single unified operator for computing $\eta_T(P,Q)^W$

Additions	$51 \cdot \frac{m-1}{2} + 503$
Multiplications	$15 \cdot \frac{m-1}{2} + 86$
Cubings	$10m + 2$
Inversion	1

A Coprocessor for the Full Pairing Computation

Results (CHES 2007)

- FPGA: Xilinx Virtex-II Pro 4
- $\mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$
- Area: 1888 slices + 6 memory blocks
- Clock frequency: 147 MHz
- Clock cycles for a full pairing: 32618
- Calculation time: 222μs

A Coprocessor for the Full Pairing Computation

Results (CHES 2007)

- FPGA: Xilinx Virtex-II Pro 4
- $\mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$
- Area: 1888 slices + 6 memory blocks
- Clock frequency: 147 MHz
- Clock cycles for a full pairing: 32618
- Calculation time: 222μs

Extended Euclidean algorithm (EEA)

- Area: 2210 additional slices
- Clock cycles for a full pairing: 32419 instead of 32618

Conclusion

Comparisons

Architecture	Area	Calculation time	FPGA
Arith 18 & Waifi 2007	18000 LEs	33 μs	Cyclone II
CHES 2007	1888 slices	222 μs	Virtex-II Pro
Grabher and Page (CHES 2005)	4481 slices	432 μs	Virtex-II Pro
Kerins *et al.* (CHES 2005)	55616 slices	850 μs	Virtex-II Pro
Ronan *et al.* (ITNG 2007)	10000 slices	178 μs	Virtex-II Pro

(1 slice $\approx$ 2 LEs)

Conclusion

VHDL code generator

- Generation of an unified operator according to $\mathbb{F}_{p^m}$ and $f(x)$
- Support for the following operations:
  - Addition
  - Multiplication
  - Frobenius ($a(x)^p \bmod f(x)$)
  - Inverse Frobenius ($\sqrt[p]{a(x)} \bmod f(x)$)

Conclusion

VHDL code generator

- Generation of an unified operator according to $\mathbb{F}_{p^m}$ and $f(x)$
- Support for the following operations:
  - Addition
  - Multiplication
  - Frobenius ($a(x)^p \bmod f(x)$)
  - Inverse Frobenius ($\sqrt[p]{a(x)} \bmod f(x)$)

Future work

- Automatic generation of the control unit
- Application (e.g. short signature)
- Genus 2
- Side-channel

Appendix

Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

$$A \cdot (-r_0^2 + y_p y_q \sigma - r_0 \rho - \rho^2) = c_0 + c_1\sigma + c_2\rho + c_3\sigma\rho + c_4\rho^2 + c_5\sigma\rho^2$$

c_0	$c_1\sigma$	$c_2\rho$	$c_3\sigma\rho$	$c_4\rho^2$	$c_5\sigma\rho^2$
$-a_4 r_0$	$-a_5 r_0$	$-a_0 r_0$	$-a_1 r_0$	$-a_2 r_0$	$-a_3 r_0$
$-a_2$	$-a_3$	$-a_4$	$-a_5$	$-a_0$	$-a_1$
		$-a_2$	$-a_3$	$-a_4$	$-a_5$
		$-a_4 r_0$	$-a_5 r_0$		
$-a_0 r_0^2$	$a_0 y_p y_q$	$-a_2 r_0^2$	$a_2 y_p y_q$	$-a_4 r_0^2$	$a_4 y_p y_q$
$-a_1 y_p y_q$	$-a_1 r_0^2$	$-a_3 y_p y_q$	$-a_3 r_0^2$	$-a_5 y_p y_q$	$-a_5 r_0^2$

Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

$$A \cdot (-r_0^2 + y_p y_q \sigma - r_0 \rho - \rho^2) = c_0 + c_1\sigma + c_2\rho + c_3\sigma\rho + c_4\rho^2 + c_5\sigma\rho^2$$

c_0	$c_1\sigma$	$c_2\rho$	$c_3\sigma\rho$	$c_4\rho^2$	$c_5\sigma\rho^2$
$-a_4 r_0$	$-a_5 r_0$	$-a_0 r_0$	$-a_1 r_0$	$-a_2 r_0$	$-a_3 r_0$
$-a_2$	$-a_3$	$-a_4$	$-a_5$	$-a_0$	$-a_1$
		$-a_2$	$-a_3$	$-a_4$	$-a_5$
		$-a_4 r_0$	$-a_5 r_0$		
$-a_0 r_0^2$	$a_0 y_p y_q$	$-a_2 r_0^2$	$a_2 y_p y_q$	$-a_4 r_0^2$	$a_4 y_p y_q$
$-a_1 y_p y_q$	$-a_1 r_0^2$	$-a_3 y_p y_q$	$-a_3 r_0^2$	$-a_5 y_p y_q$	$-a_5 r_0^2$

1. Compute in parallel r_0^2, $y_p y_q$, $a_0 r_0$, $a_1 r_0$, $a_2 r_0$, $a_3 r_0$, $a_4 r_0$, and $a_5 r_0$ (8 multiplications)

Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

$$A \cdot (-r_0^2 + y_p y_q \sigma - r_0 \rho - \rho^2) = c_0 + c_1\sigma + c_2\rho + c_3\sigma\rho + c_4\rho^2 + c_5\sigma\rho^2$$

c_0	$c_1\sigma$	$c_2\rho$	$c_3\sigma\rho$	$c_4\rho^2$	$c_5\sigma\rho^2$
$-a_4 r_0$	$-a_5 r_0$	$-a_0 r_0$	$-a_1 r_0$	$-a_2 r_0$	$-a_3 r_0$
$-a_2$	$-a_3$	$-a_4$	$-a_5$	$-a_0$	$-a_1$
		$-a_2$	$-a_3$	$-a_4$	$-a_5$
		$-a_4 r_0$	$-a_5 r_0$		
$-a_0 r_0^2$	$a_0 y_p y_q$	$-a_2 r_0^2$	$a_2 y_p y_q$	$-a_4 r_0^2$	$a_4 y_p y_q$
$-a_1 y_p y_q$	$-a_1 r_0^2$	$-a_3 y_p y_q$	$-a_3 r_0^2$	$-a_5 y_p y_q$	$-a_5 r_0^2$

1. Compute in parallel r_0^2, $y_p y_q$, $a_0 r_0$, $a_1 r_0$, $a_2 r_0$, $a_3 r_0$, $a_4 r_0$, and $a_5 r_0$ (8 multiplications)
2. Apply Karatsuba's algorithm to compute the remaining products by means of 9 multipliers

Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

$$A \cdot (-r_0^2 + y_p y_q \sigma - r_0 \rho - \rho^2) = c_0 + c_1 \sigma + c_2 \rho + c_3 \sigma \rho + c_4 \rho^2 + c_5 \sigma \rho^2$$

$-a_0 r_0^2$	$a_0 y_p y_q$	$-a_2 r_0^2$	$a_2 y_p y_q$	$-a_4 r_0^2$	$a_4 y_p y_q$
$-a_1 y_p y_q$	$-a_1 r_0^2$	$-a_3 y_p y_q$	$-a_3 r_0^2$	$-a_5 y_p y_q$	$-a_5 r_0^2$

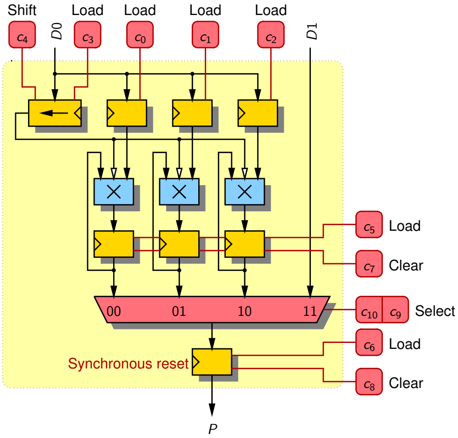
Karatsuba's algorithm (9 multiplications performed in parallel):

$$a_0 y_p y_q - a_1 r_0^2 = (a_0 + a_1) \times (y_p y_q - r_0^2) + a_0 \times r_0^2 - a_1 \times y_p y_q$$
$$a_2 y_p y_q - a_3 r_0^2 = (a_2 + a_3) \times (y_p y_q - r_0^2) + a_2 \times r_0^2 - a_3 \times y_p y_q$$
$$a_4 y_p y_q - a_5 r_0^2 = (a_4 + a_5) \times (y_p y_q - r_0^2) + a_4 \times r_0^2 - a_5 \times y_p y_q$$

Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

M_0	M_1	M_2
$a_0 r_0$	$a_2 r_0$	$a_4 r_0$
$a_0 r_0^2$	$a_2 r_0^2$	$a_4 r_0^2$

- Three multipliers
- Common operand: r_0 or r_0^2

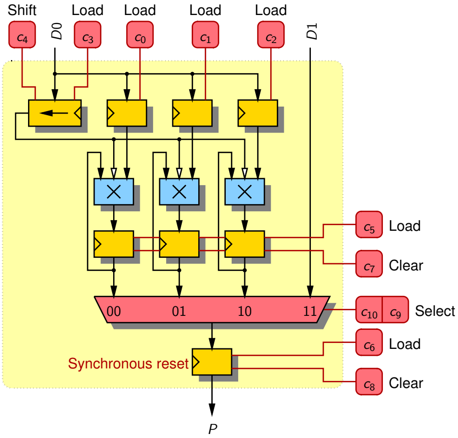


Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P, Q)$

M_3	M_4	M_5
$a_1 r_0$	$a_3 r_0$	$a_5 r_0$
$a_1 y_p y_q$	$a_3 y_p y_q$	$a_5 y_p y_q$

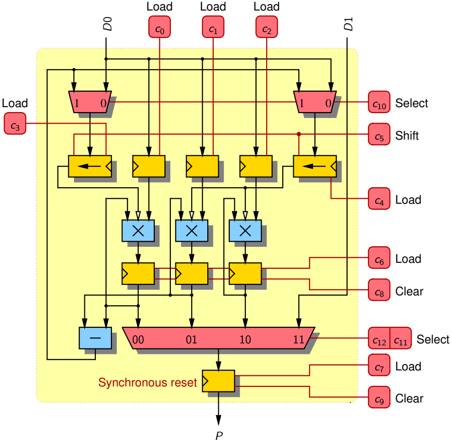
- Three multipliers
- Common operand:

  r_0 or $y_p y_q$

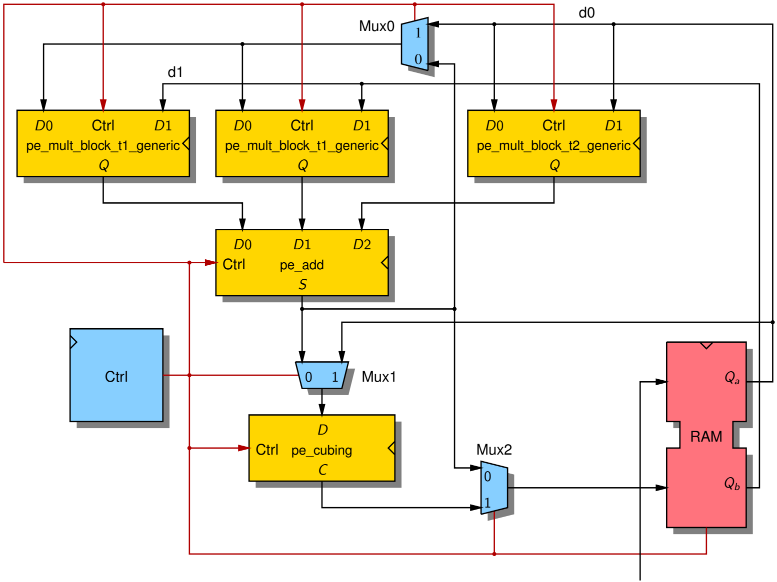


Multiplication over $\mathbb{F}_{3^{6m}}$ – $\eta_T(P,Q)$

M_6	M_7	M_8
r_0^2	$y_p y_q$	–
$(a_0 + a_1) \times (y_p y_q - r_0^2)$	$(a_2 + a_3) \times (y_p y_q - r_0^2)$	$(a_4 + a_5) \times (y_p y_q - r_0^2)$



A Coprocessor for the η_T Pairing Computation



A Coprocessor for the Full Pairing Computation