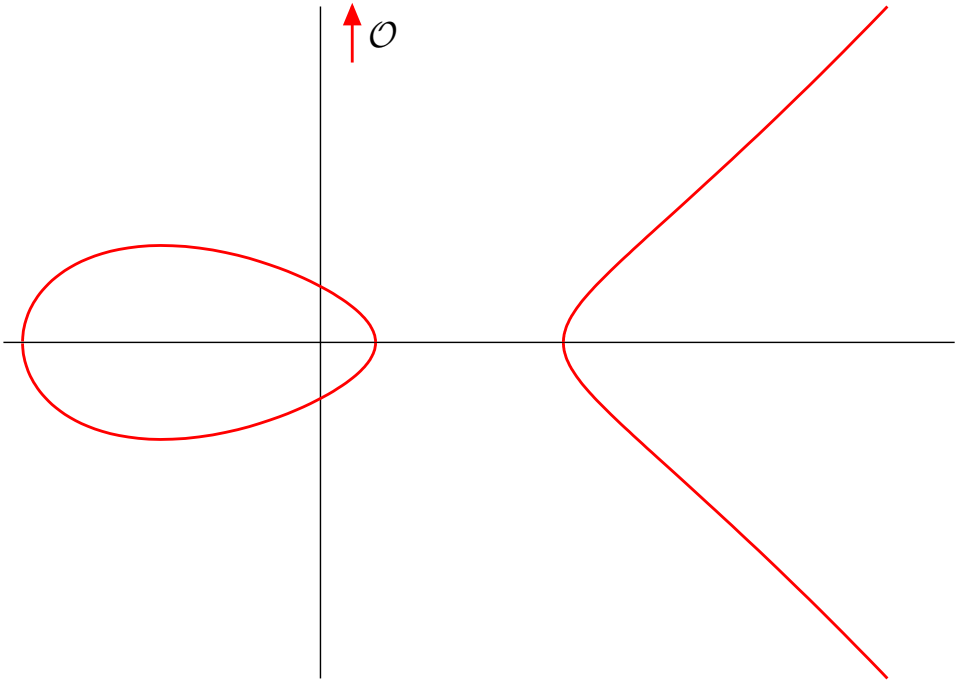
Cairn seminar — November 29th, 2013

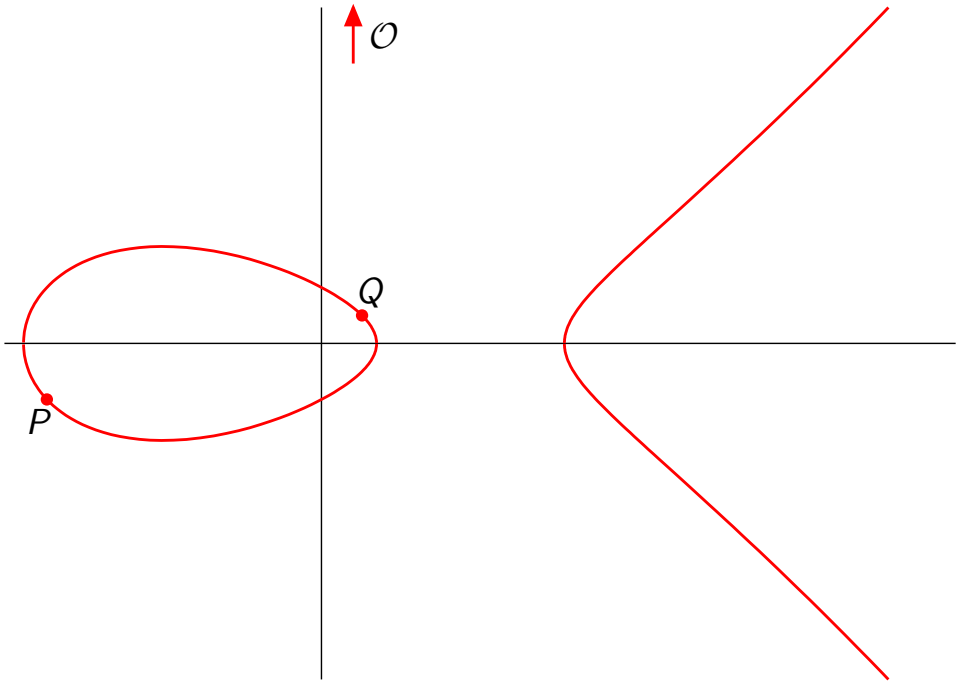
Algorithms and arithmetic for the implementation of cryptographic pairings

Nicolas Estibals
CAIRN project-team, IRISA
Nicolas.Estibals@irisa.fr

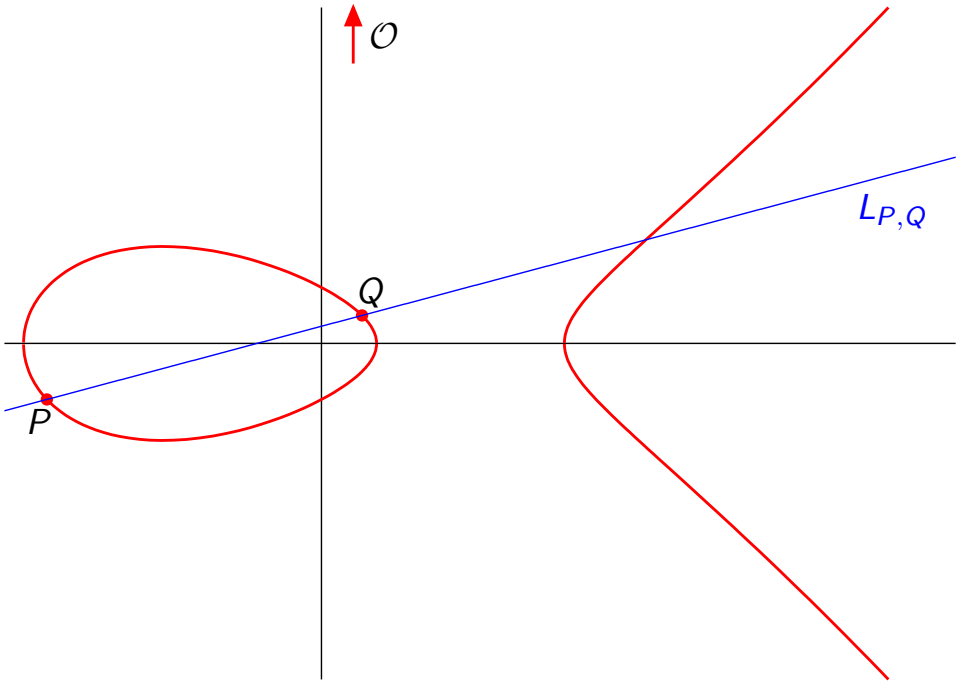
What is an elliptic curve?



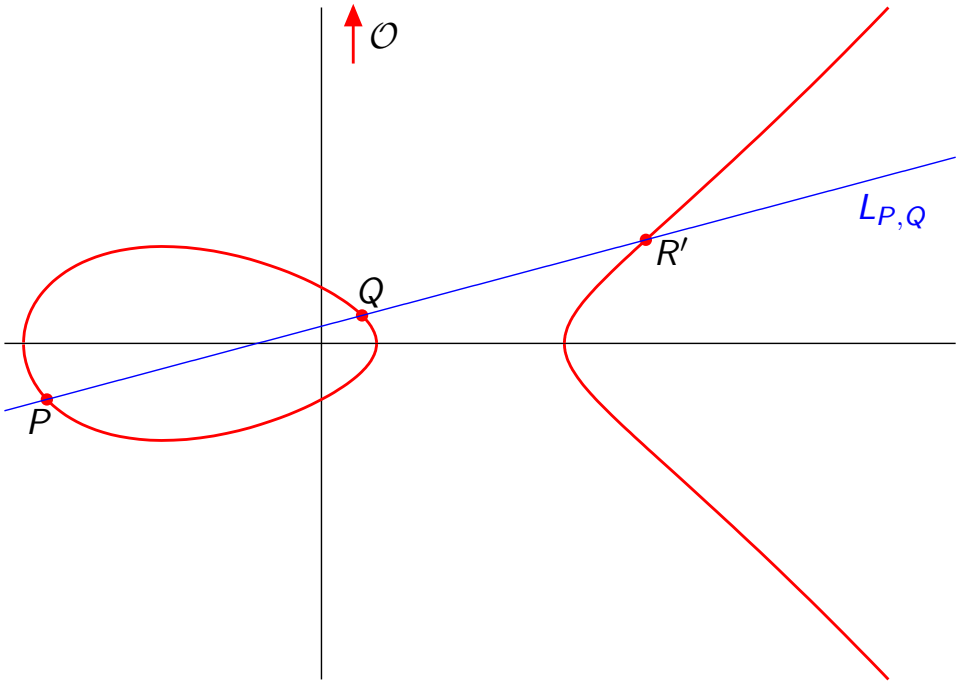
What is an elliptic curve?



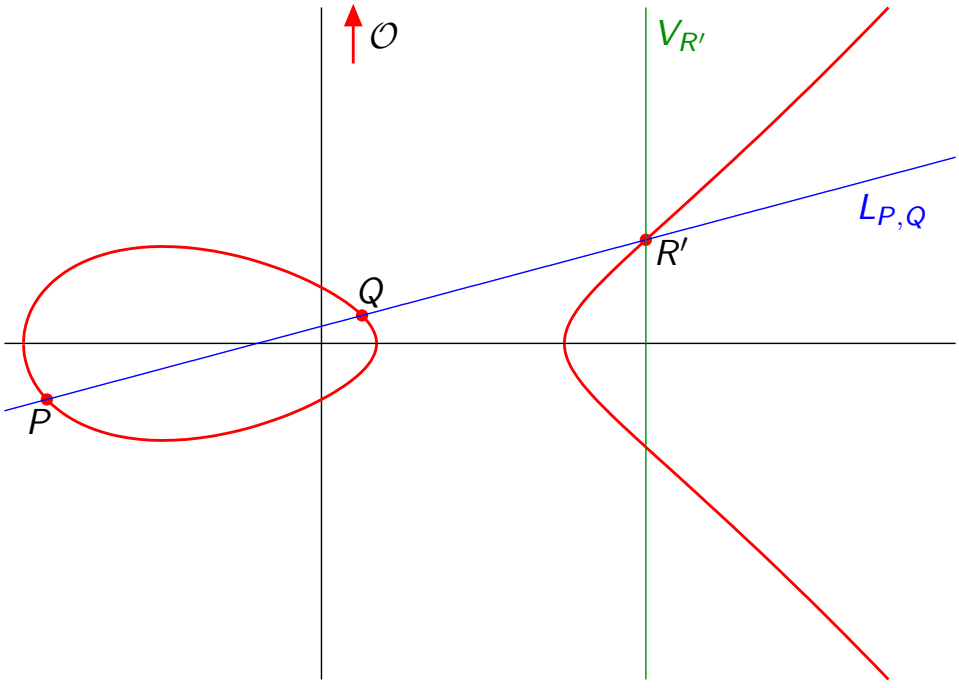
What is an elliptic curve?



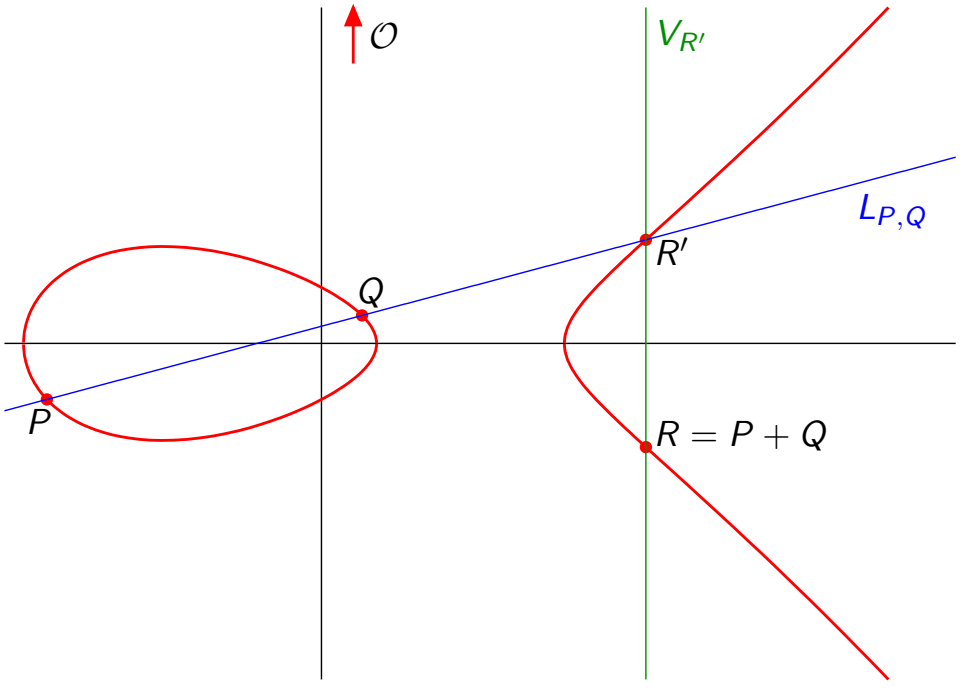
What is an elliptic curve?



What is an elliptic curve?



What is an elliptic curve?



Elliptic Curve Cryptography

Discrete Logarithm Problem (DLP)

Let $\mathbb{G}$ be a cyclic group, P a generator, given $Q \in \mathbb{G}$, it is supposed to be hard to compute a such that

$$Q = [a]P$$

Elliptic Curve Cryptography

Discrete Logarithm Problem (DLP)

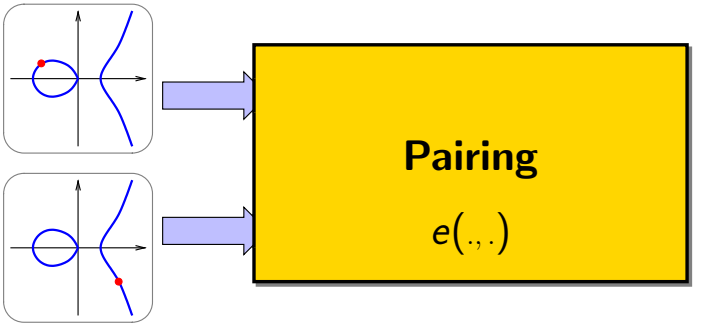
Let $\mathbb{G}$ be a cyclic group, P a generator, given $Q \in \mathbb{G}$, it is supposed to be hard to compute a such that

$$Q = [a]P$$

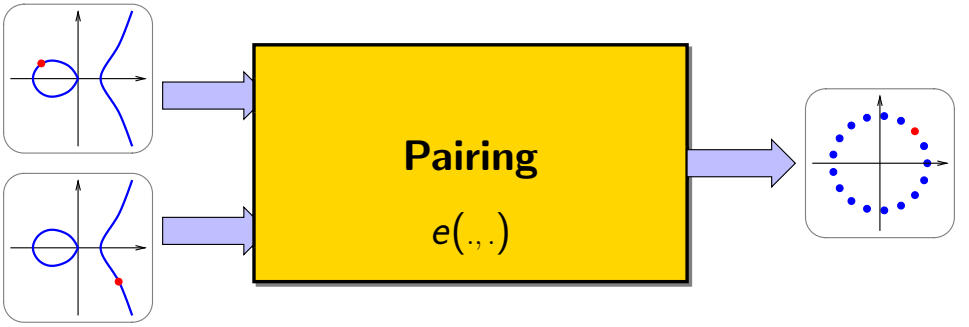
- Use this hard problem to design cryptographic protocols
- Diffie–Hellman key exchange:
  - Alice generates a secret integer a
  - Alice sends $[a]P$ to Bob
  - Alice computes $[a][b]P$
  - Bob generates a secret integer b
  - Bob sends $[b]P$ to Alice
  - Bob computes $[b][a]P$

They both share the same secret: $[ab]P$

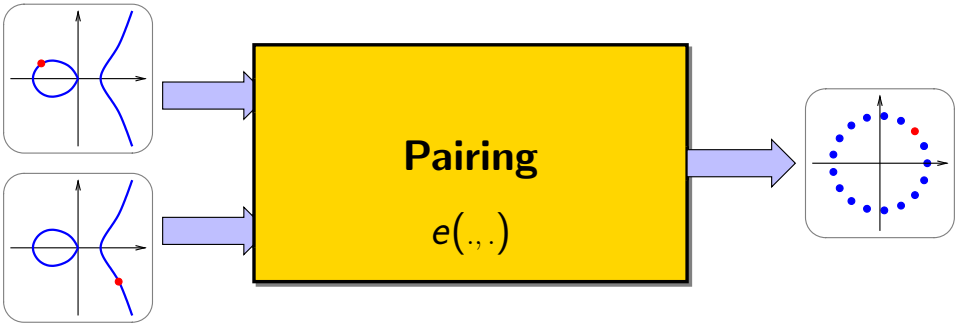
What is a pairing?



What is a pairing?



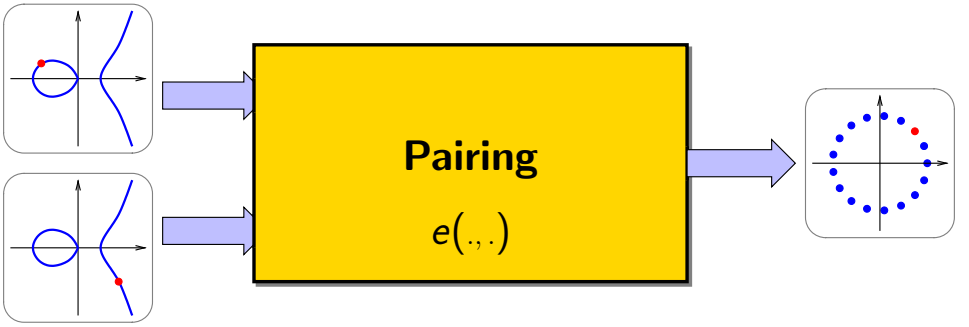
What is a pairing?



- Cryptographic interest: *Mixing two secrets without having to know them*

$$e([a]P, [b]Q) = e(P, Q)^{ab}$$

What is a pairing?

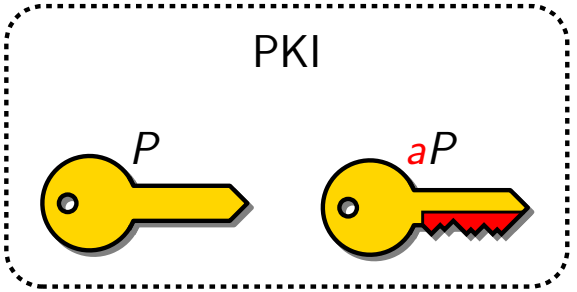


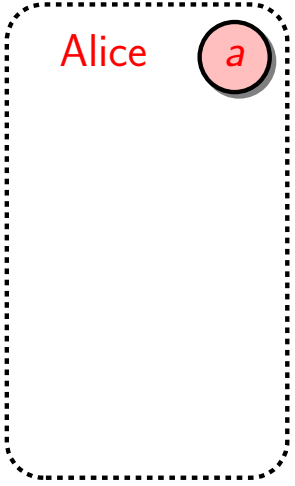
- Cryptographic interest: *Mixing two secrets without having to know them*

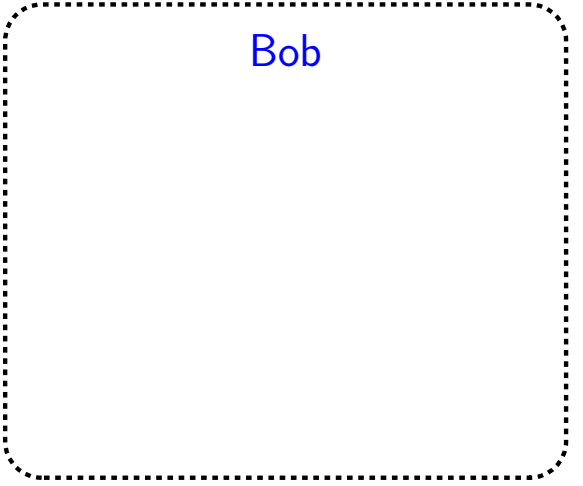
$$e([a]P, [b]Q) = e(P, Q)^{ab}$$

- Useful for advanced protocols
  - short signature
  - electronic voting
  - electronic money
  - ...

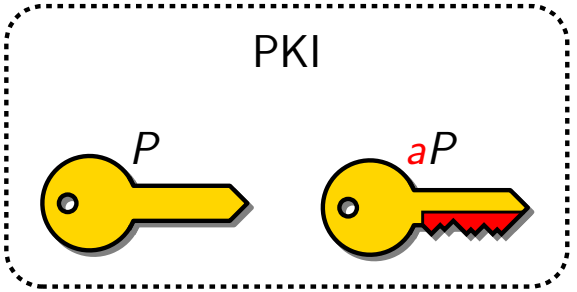
Example: Short signature

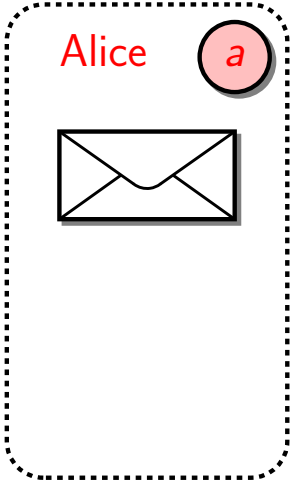


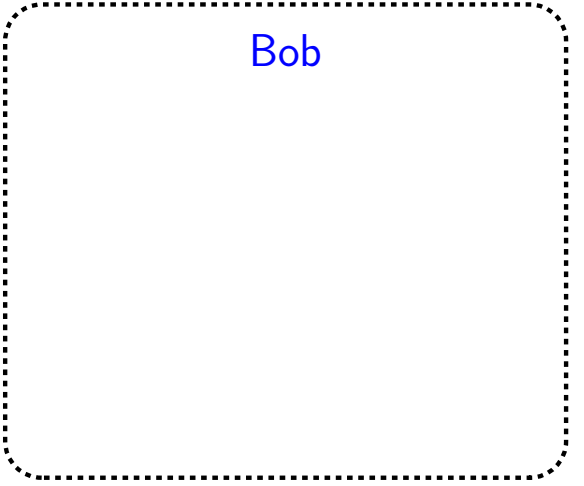




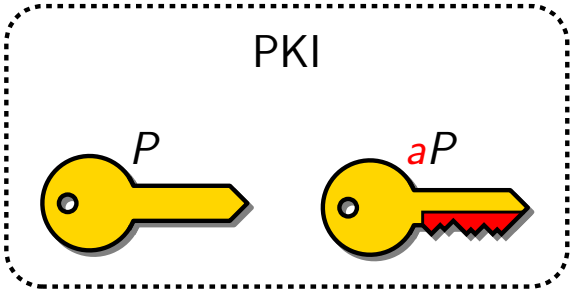
Example: Short signature

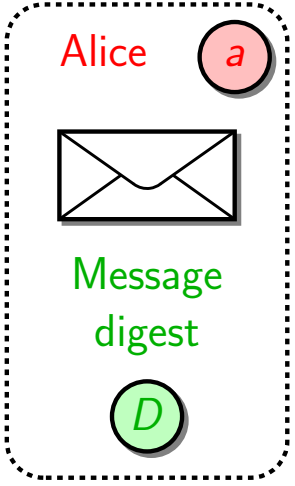


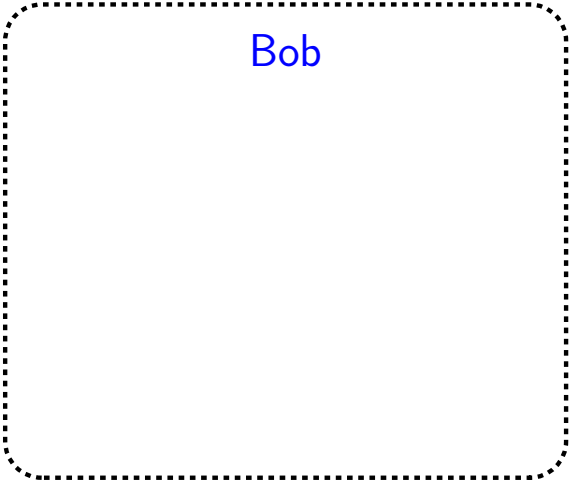




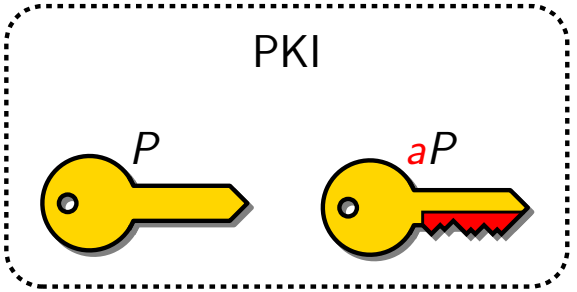
Example: Short signature

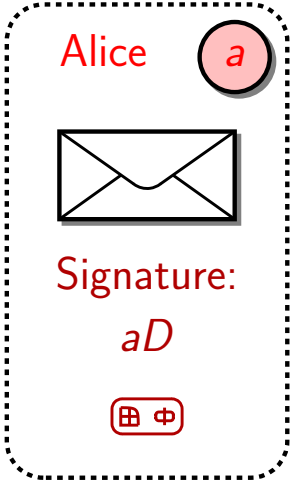


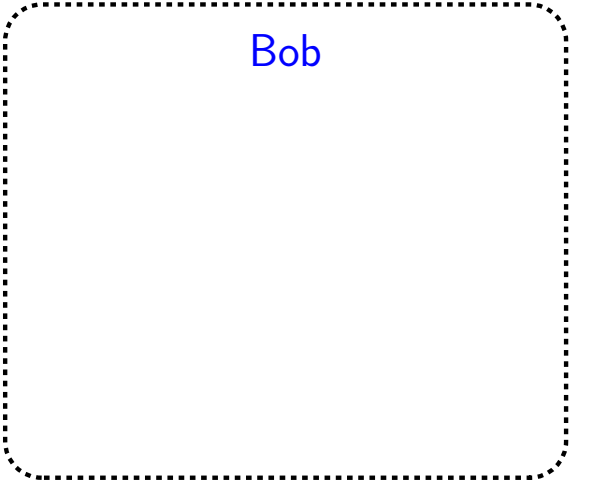




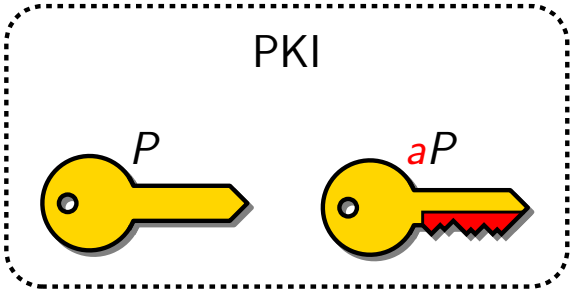
Example: Short signature

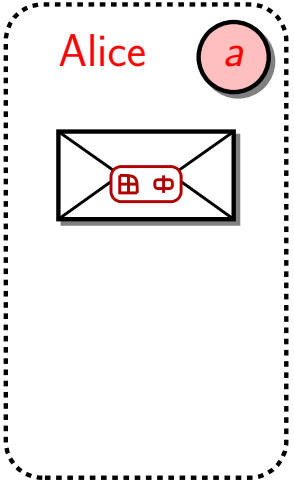


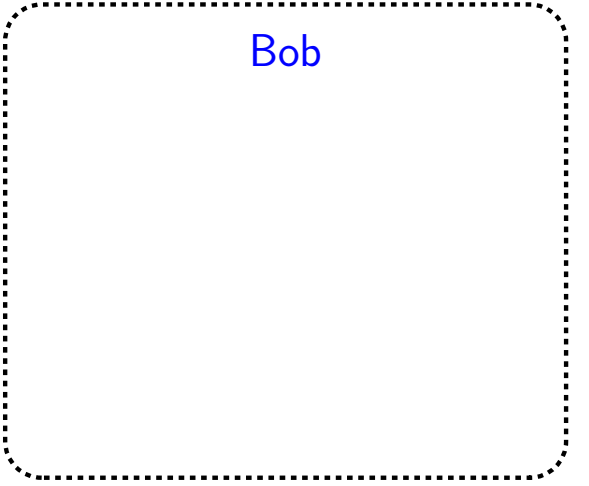




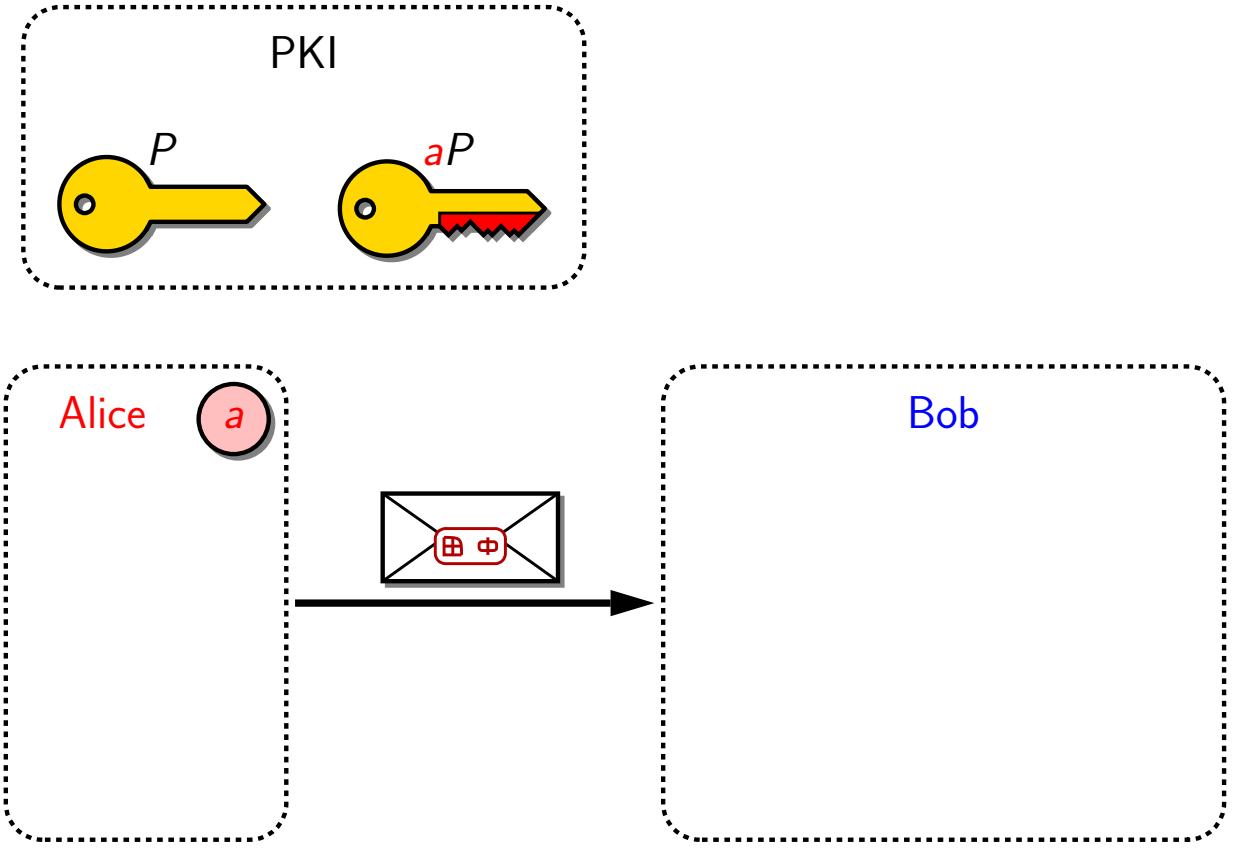
Example: Short signature



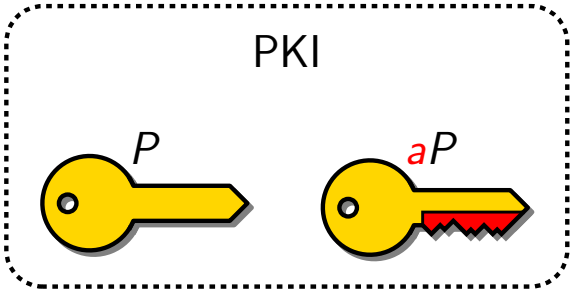


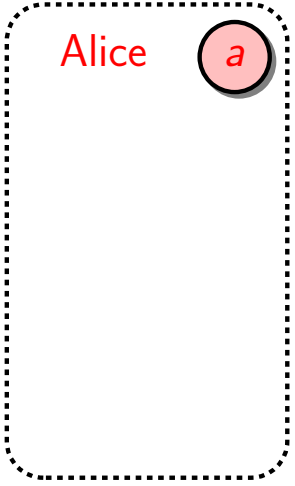


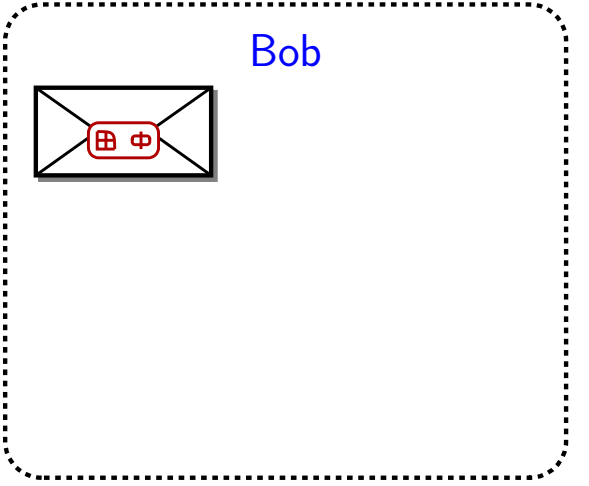
Example: Short signature



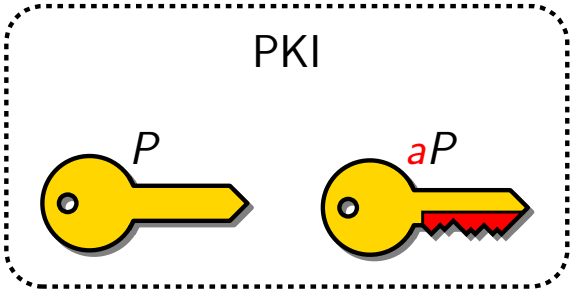
Example: Short signature

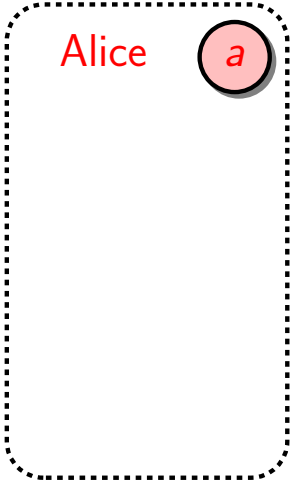


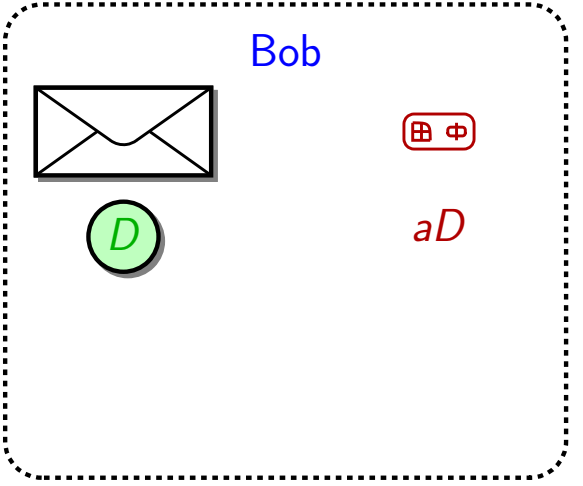




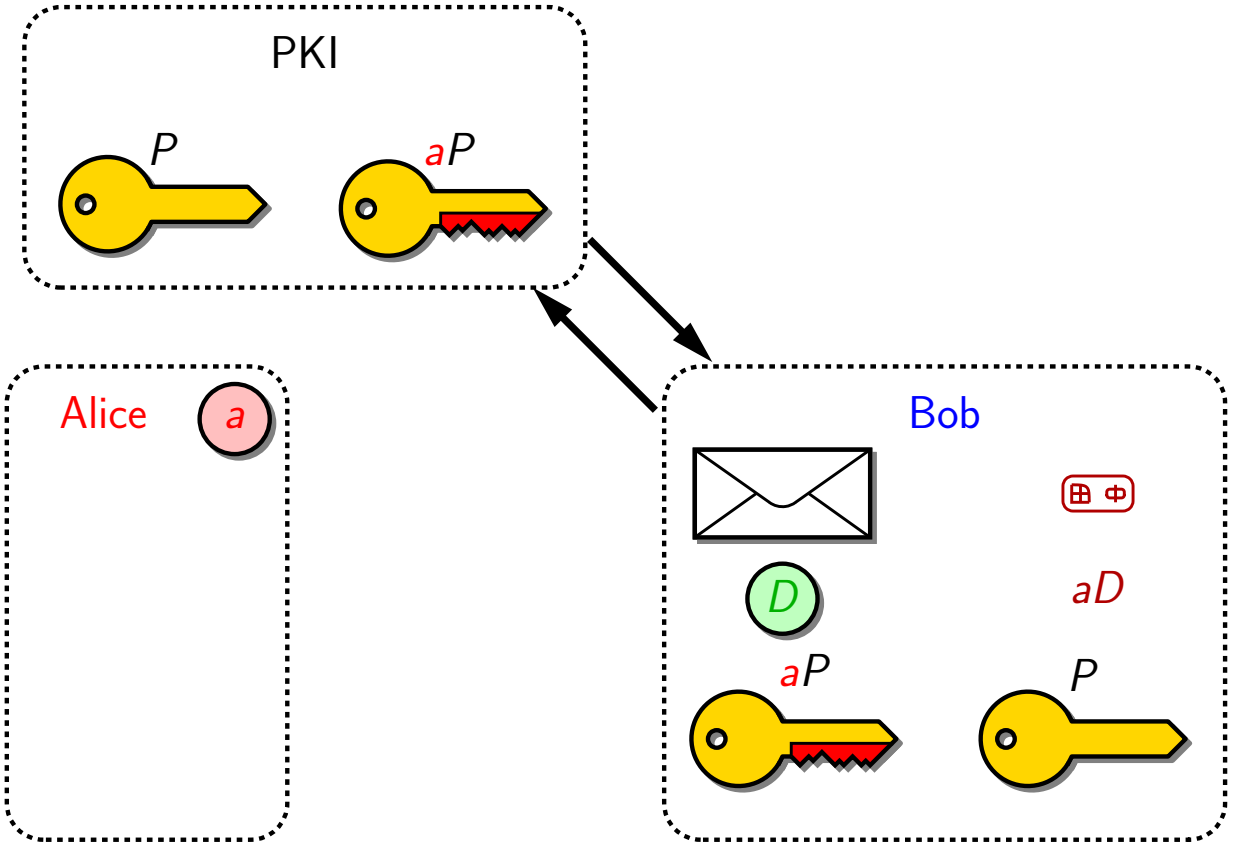
Example: Short signature



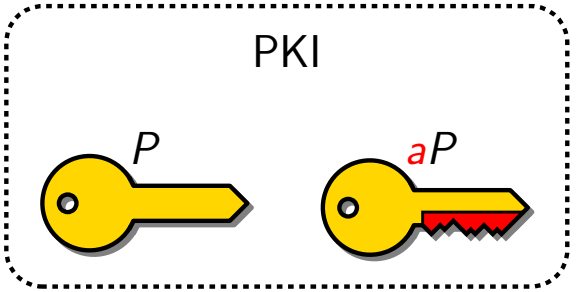


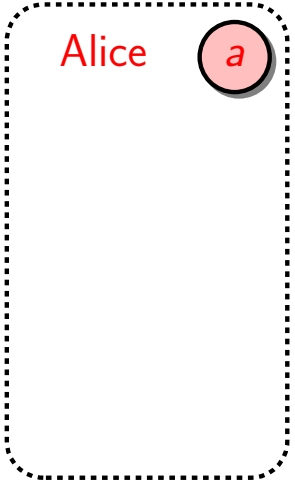


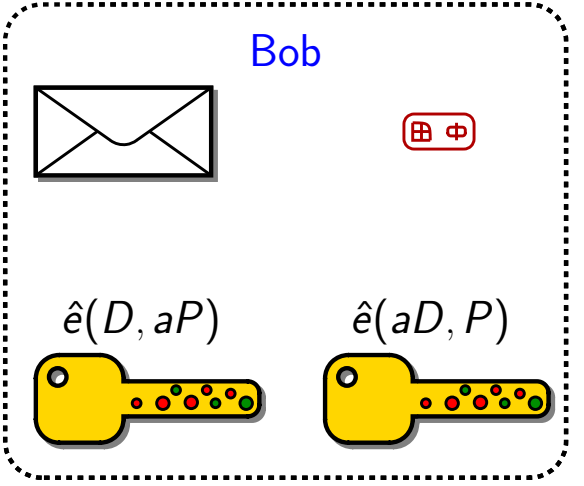
Example: Short signature



Example: Short signature







Security considerations

- DLP should be hard on all the groups involved

Security considerations

- DLP should be hard on all the groups involved
- Security measurement
  - number of operations to break a cryptosystem
  - today's recommendation: 128-bit security

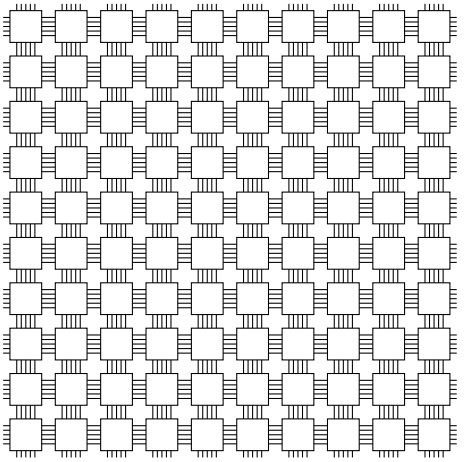
$$2^{128} \text{ operations}$$

Why cryptography and hardware implementations?

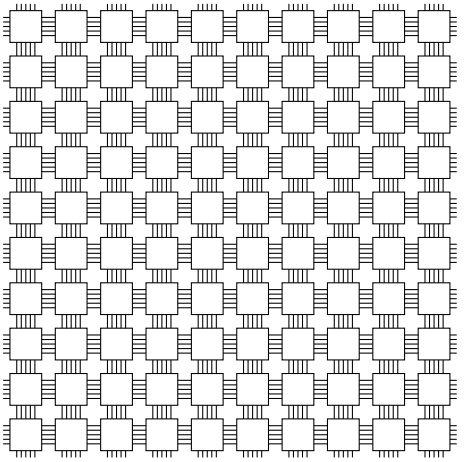
- Growth of numeric exchanges
  - many applications
    - ⋆ bank services
    - ⋆ secure firmware updates
    - ⋆ personal communications
    - ⋆ ...
  - many targets
    - ⋆ embedded electronics
    - ⋆ smart cards
    - ⋆ smartphones
    - ⋆ computers, servers
- Security implies non-trivial computations
- Need for hardware implementations
  - CPUs may be inadequate
  - limited resources

Hardware implementation

- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection

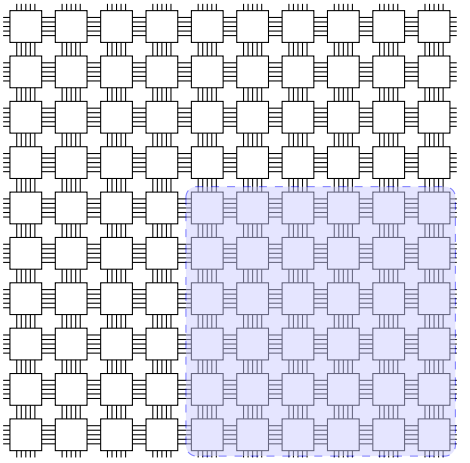


Hardware implementation



- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)

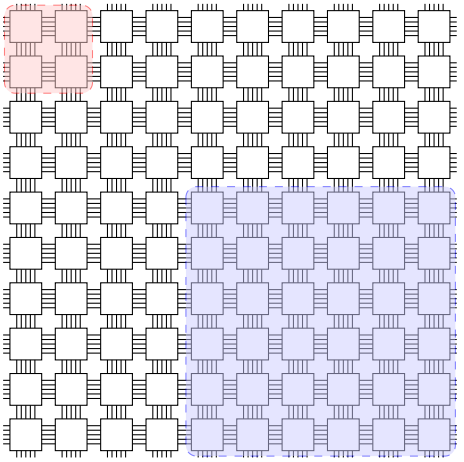
Hardware implementation

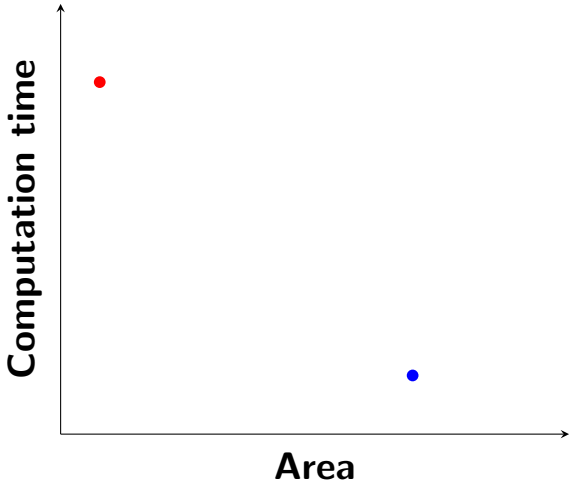


- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)
  - area (slices)

Hardware implementation

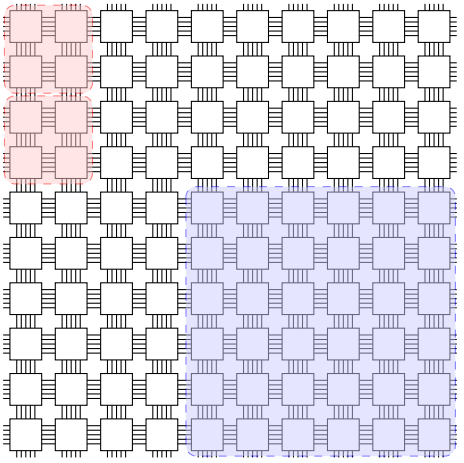
- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)
  - area (slices)
- Different designs for the same computation
  - optimized for latency
  - optimized for compactness

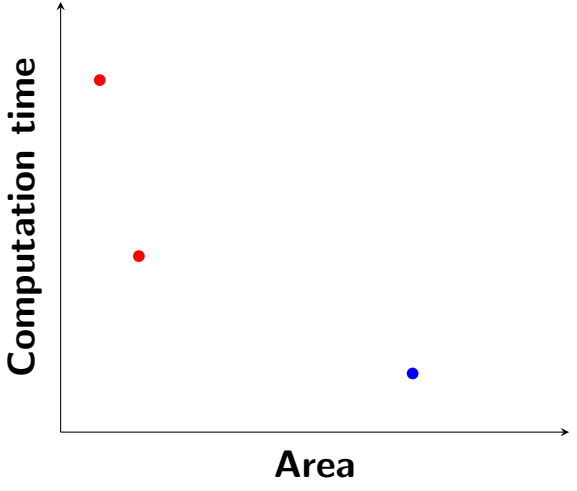




Hardware implementation

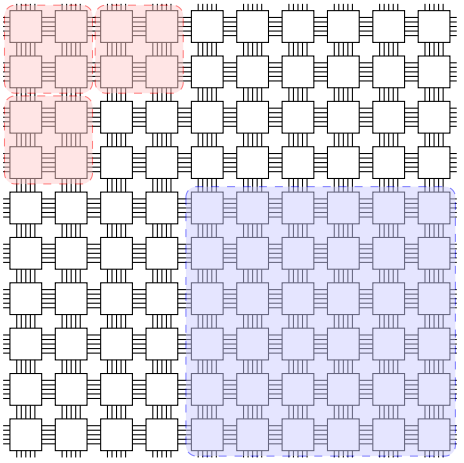
- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)
  - area (slices)
- Different designs for the same computation
  - optimized for latency
  - optimized for compactness

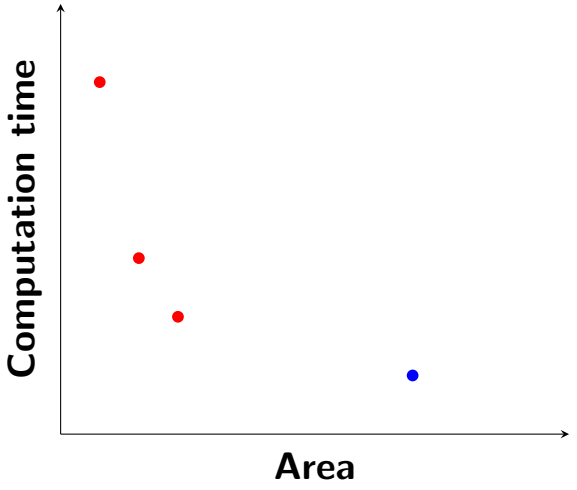




Hardware implementation

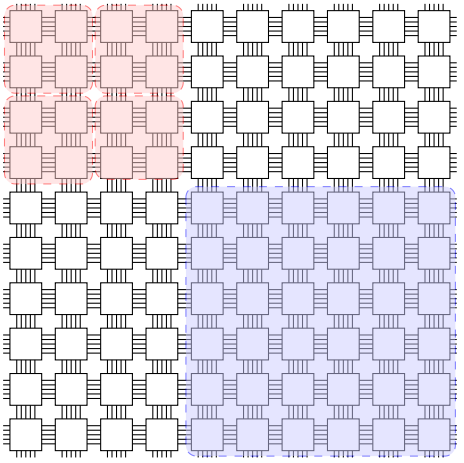
- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)
  - area (slices)
- Different designs for the same computation
  - optimized for latency
  - optimized for compactness

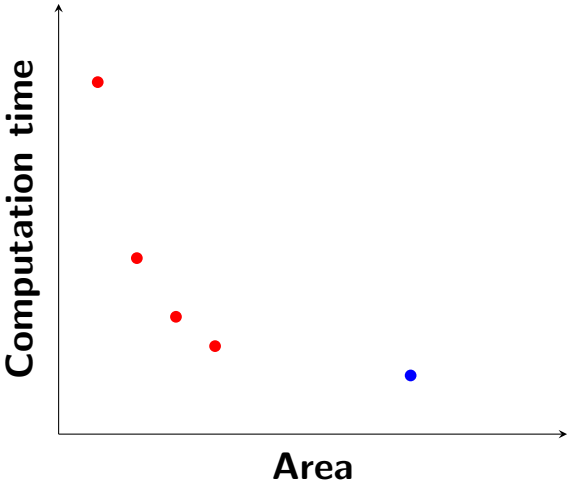




Hardware implementation

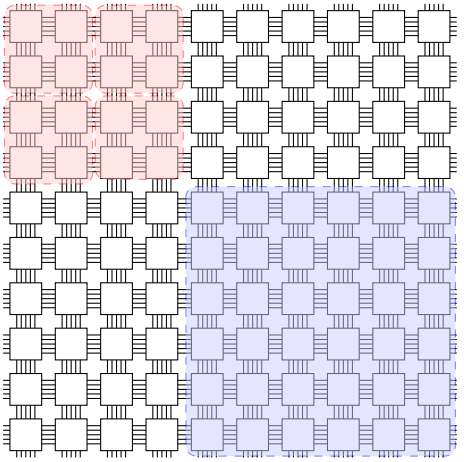
- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)
  - area (slices)
- Different designs for the same computation
  - optimized for latency
  - optimized for compactness

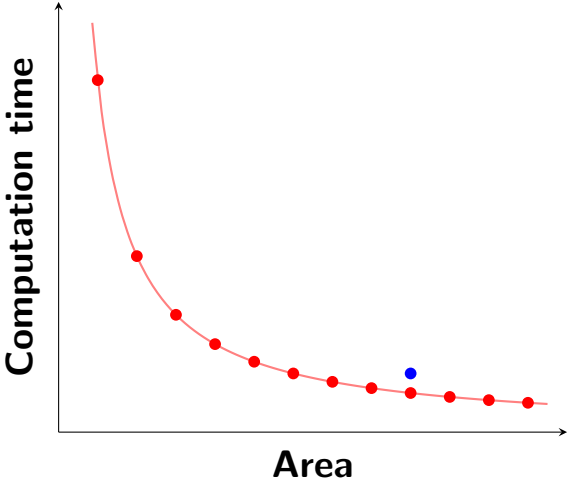




Hardware implementation

- Our target: Field Programmable Gate Array (FPGA)
  - integrated circuit
  - matrix of simple configurable logic cells
  - programmable interconnection
- Performance metric
  - time (ms)
  - area (slices)
  - time–area product
- Different designs for the same computation
  - optimized for latency
  - optimized for compactness
  - optimized for throughput





Contributions

- Fast accelerator for pairings [CHES 2009, IEEE TC 2011]
  Joint work with Beuchat, Detrey, Okamoto and Rodríguez-Henríquez
  - parallel architecture
  - pipelined subquadratic multiplier
- Compact design for pairings reaching 128-bit security
  - composite extension fields [Paring 2010]
  - hyperelliptic curves [CT-RSA 2012]
    Joint work with Aranha, Beuchat and Detrey
- Formulae for sub-quadratic multiplication [WAIFI 2012]
  Joint work with Barbulescu, Detrey and Zimmermann
  - exhaustive search
  - improved formulae for $\mathbb{F}_{3^{5m}}$

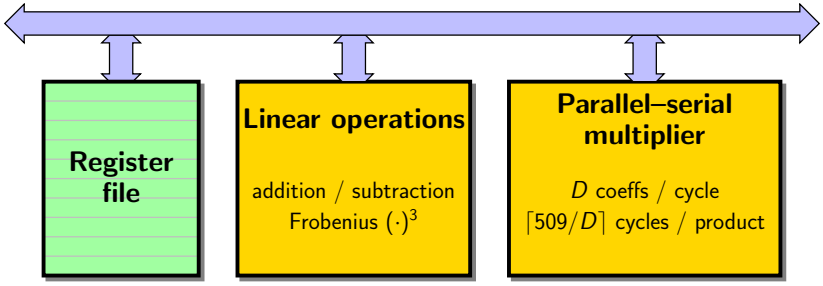
Contributions

- Fast accelerator for pairings [CHES 2009, IEEE TC 2011]
  Joint work with Beuchat, Detrey, Okamoto and Rodríguez-Henríquez
  - parallel architecture
  - pipelined subquadratic multiplier
- Compact design for pairings reaching 128-bit security
  - composite extension fields [Paring 2010]
  - hyperelliptic curves [CT-RSA 2012]
    Joint work with Aranha, Beuchat and Detrey
- Formulae for sub-quadratic multiplication [WAIFI 2012]
  Joint work with Barbulescu, Detrey and Zimmermann
  - exhaustive search
  - improved formulae for $\mathbb{F}_{3^{5m}}$

Outline of the talk

- Compact design through composite extension fields
- Pipelined subquadratic multiplier
- Conclusion and Perspectives

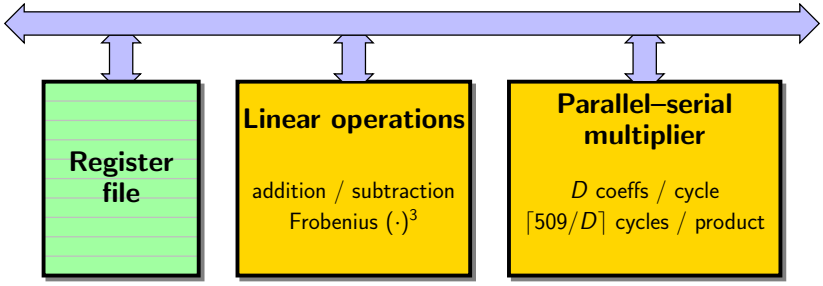
An arithmetic coprocessor



- For a supersingular elliptic curve over $\mathbb{F}_{3^{509}}$
- Only need arithmetic operations in $\mathbb{F}_{3^{509}}$
  - implement a specialized processor
- Multiplication is critical
  - separate linear operations and multiplications
  - careful scheduling to keep multiplier busy

Operation	count
$\times$	3638
$+$	17240
$(.)^3$	4068
$(.)^{-1}$	1

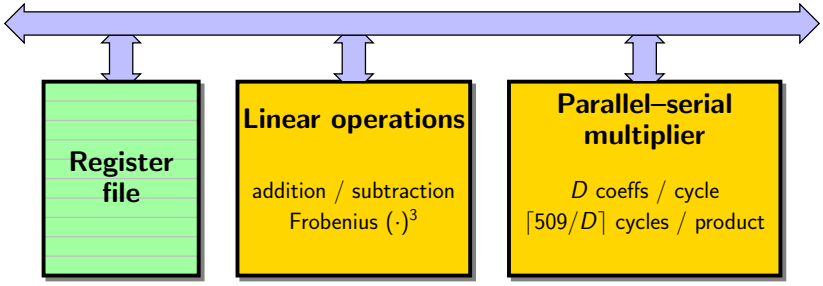
An arithmetic coprocessor



- For a supersingular elliptic curve over $\mathbb{F}_{3^{509}}$
- Only need arithmetic operations in $\mathbb{F}_{3^{509}}$
  - implement a specialized processor
- Multiplication is critical
  - separate linear operations and multiplications
  - careful scheduling to keep multiplier busy
- Inverse is only needed once: Itoh–Tsujii algorithm
  - no need for hardware support

Operation	count
$\times$	3638
$+$	17240
$(.)^3$	4068
$(.)^{-1}$	1

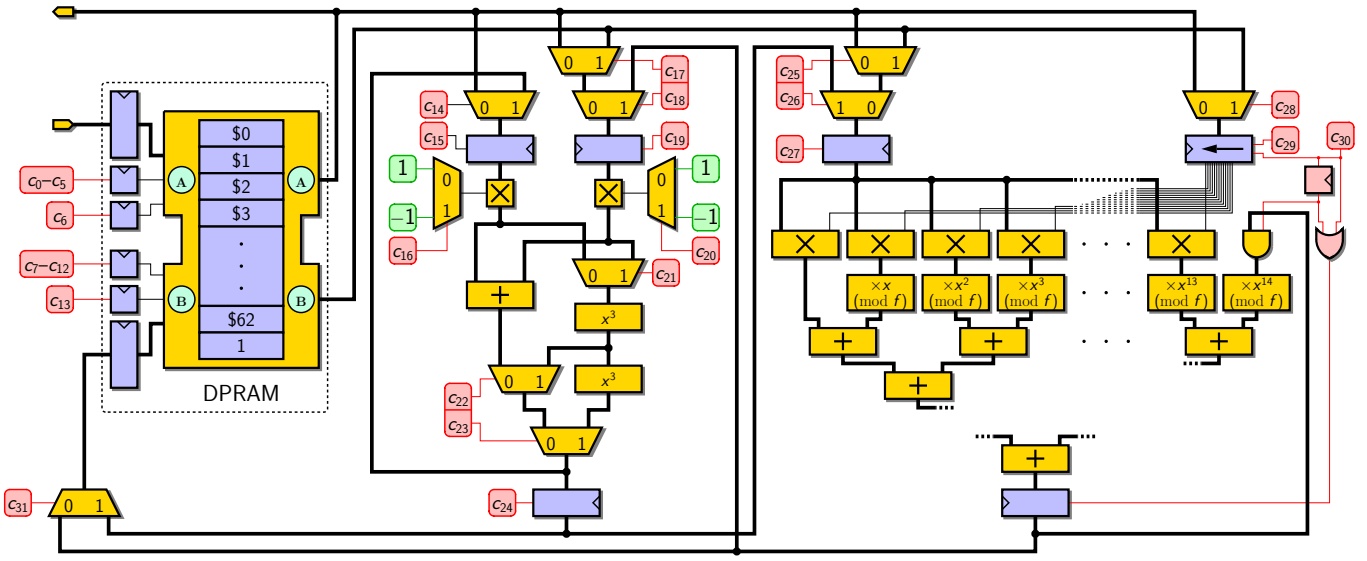
An arithmetic coprocessor



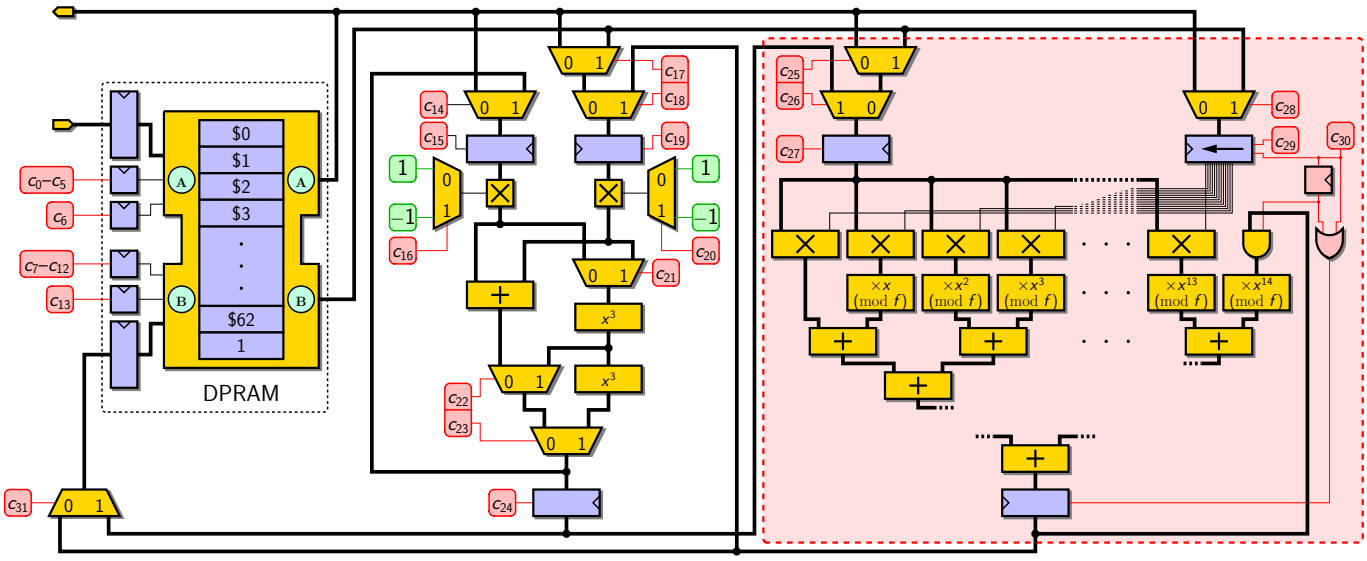
- For a supersingular elliptic curve over $\mathbb{F}_{3^{509}}$
- Only need arithmetic operations in $\mathbb{F}_{3^{509}}$
  - implement a specialized processor
- Multiplication is critical
  - separate linear operations and multiplications
  - careful scheduling to keep multiplier busy
- Inverse is only needed once: Itoh–Tsujii algorithm
  - no need for hardware support
- Synthesis results for $\mathbb{F}_{3^{509}}$: 9625 slices
  - almost fully occupy a Virtex 6 LX 75 T (82%)
  - computation time: ≈ 4 ms

Operation	count
$\times$	3638
$+$	17240
$(.)^3$	4068
$(.)^{-1}$	1

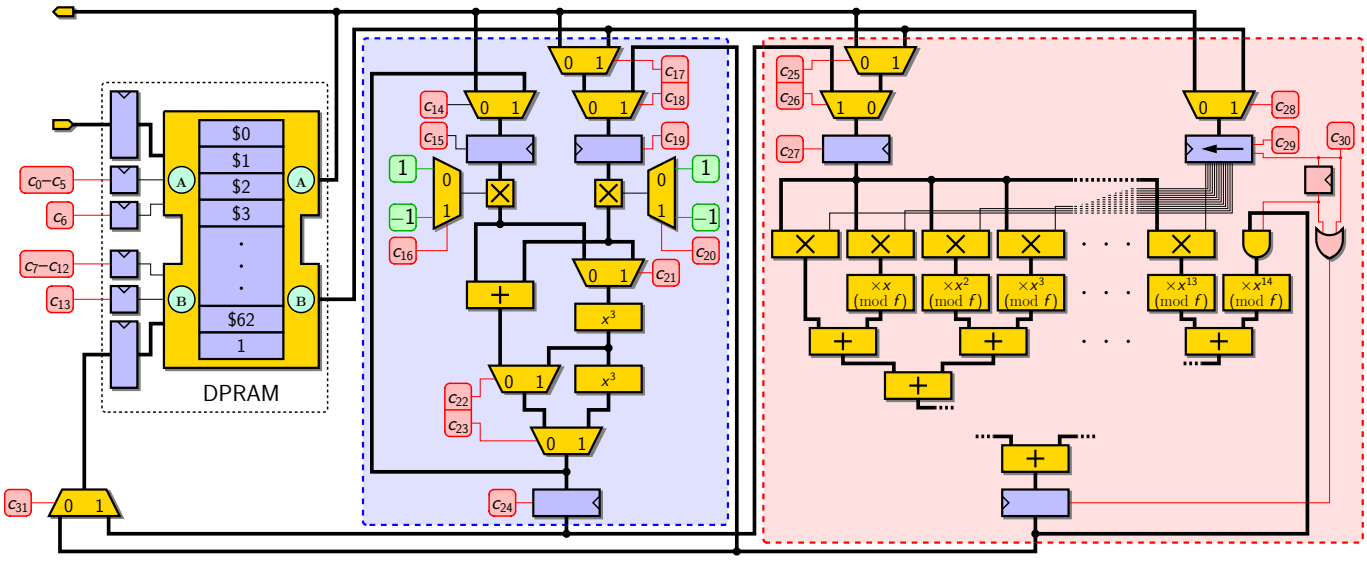
Detailed architecture of the coprocessor (char. 3)



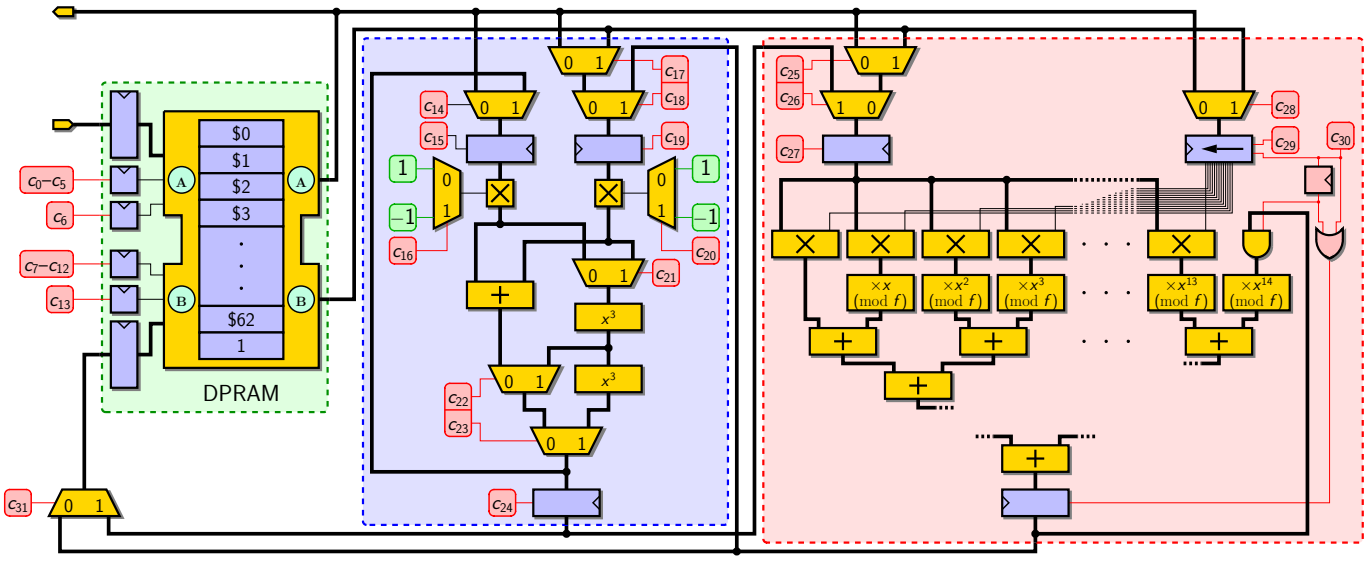
Detailed architecture of the coprocessor (char. 3)



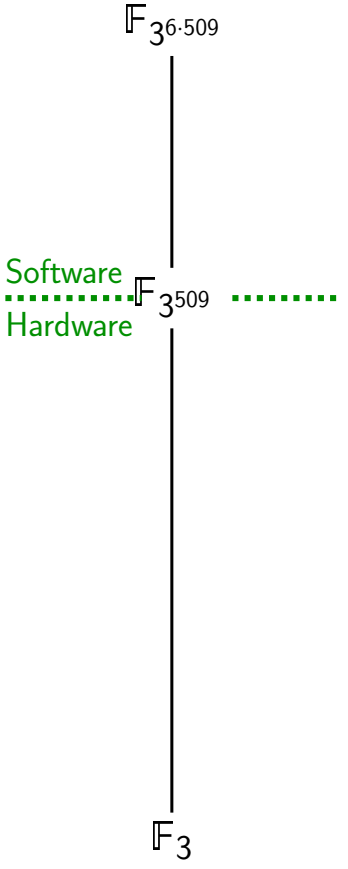
Detailed architecture of the coprocessor (char. 3)



Detailed architecture of the coprocessor (char. 3)

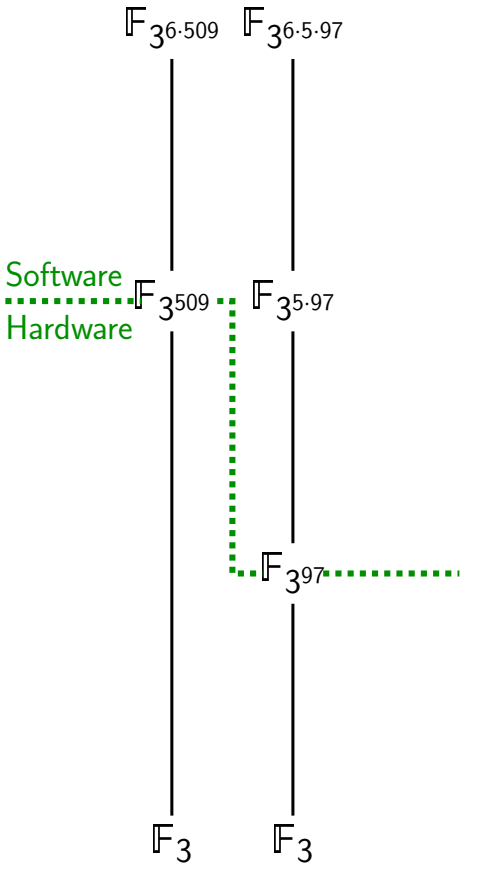


Field of composite extension degree



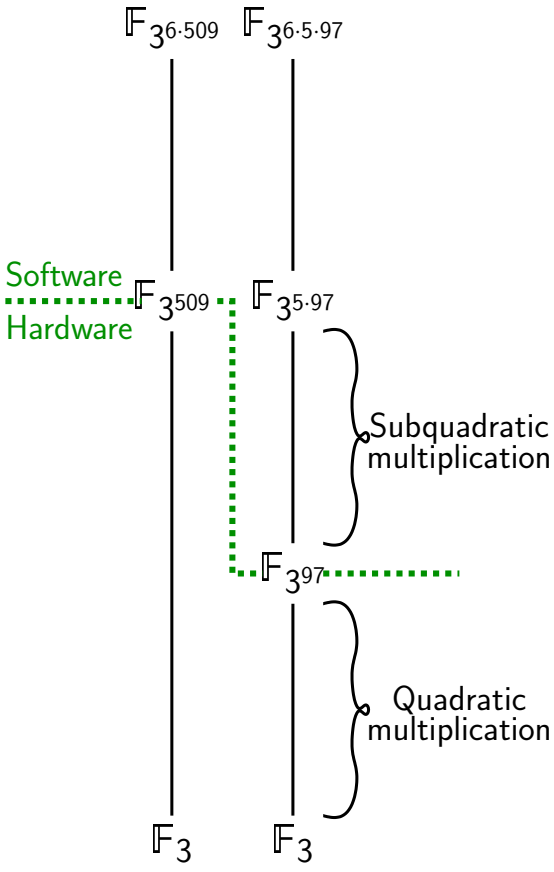
Field of composite extension degree

- Provides some arithmetic advantages
  - smaller datapath



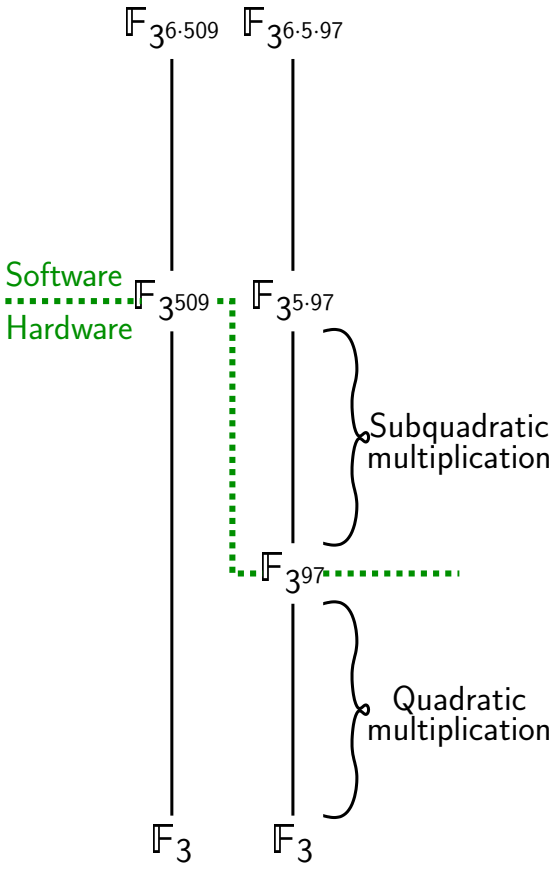
Field of composite extension degree

- Provides some arithmetic advantages
  - smaller datapath
  - efficient multiplication algorithm



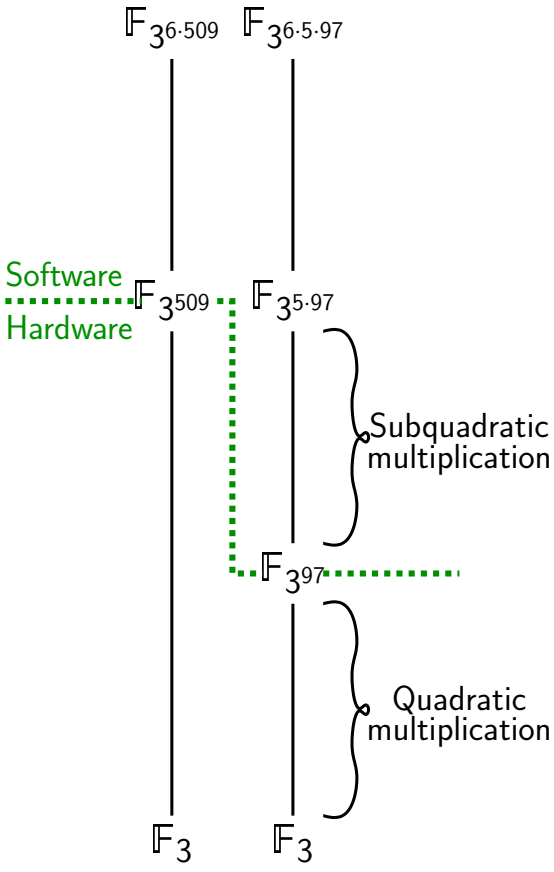
Field of composite extension degree

- Provides some arithmetic advantages
  - smaller datapath
  - efficient multiplication algorithm
- Allows supplementary attacks on the curve
  - with limited effect on security

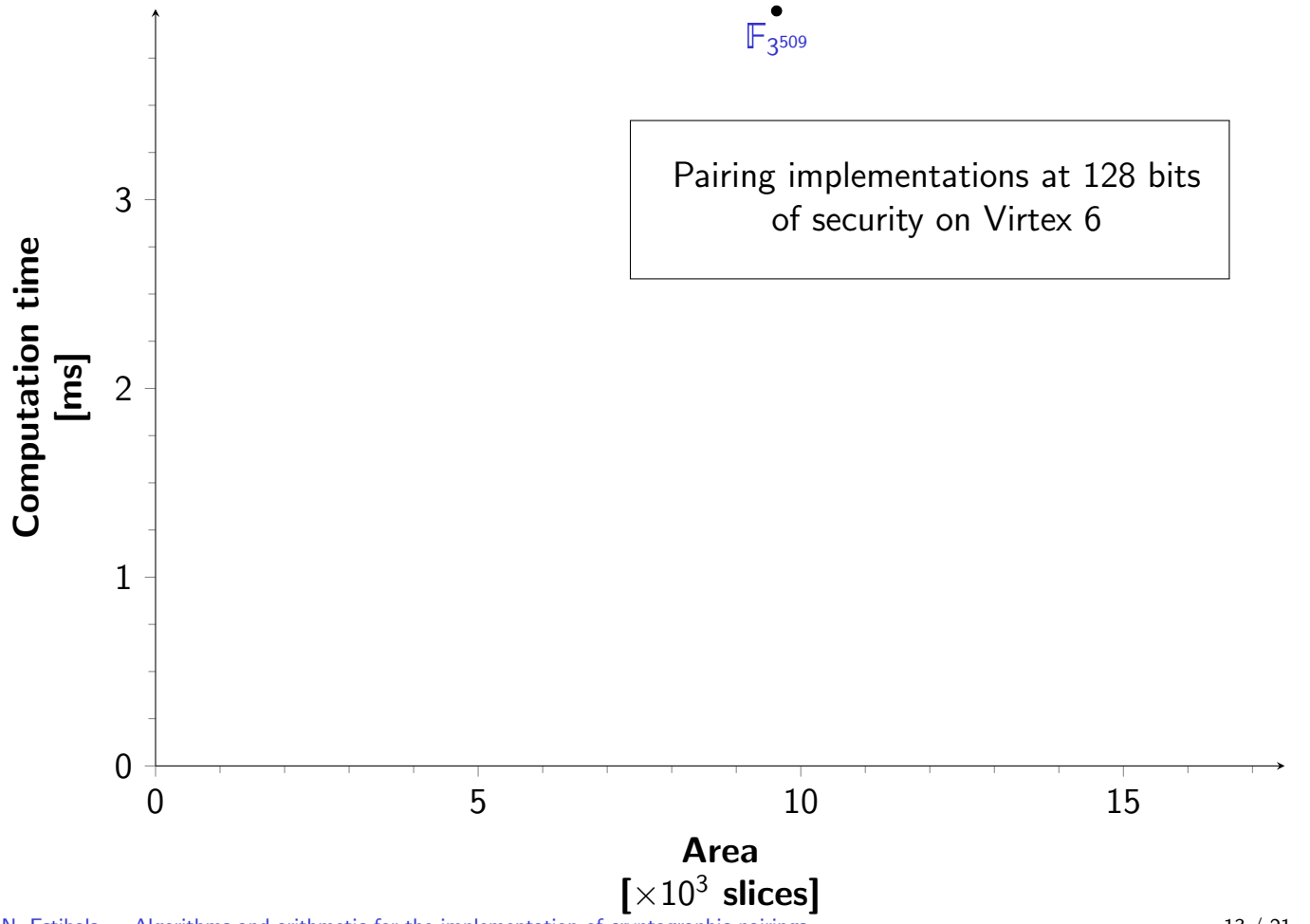


Field of composite extension degree

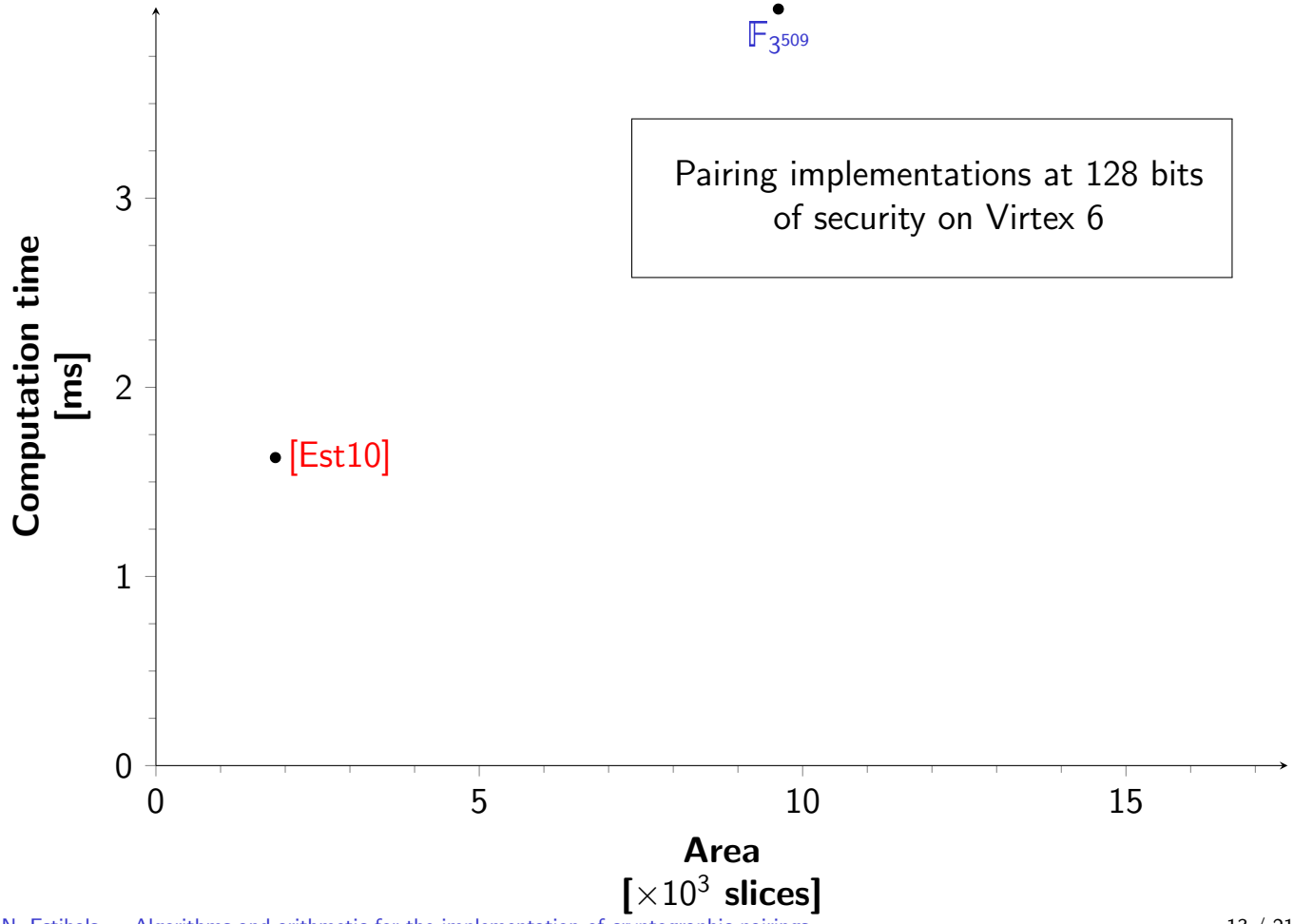
- Provides some arithmetic advantages
  - smaller datapath
  - efficient multiplication algorithm
- Allows supplementary attacks on the curve
  - with limited effect on security
- Results
  - 1848 slices of the same Virtex 6 LX (15%)
    5.2 times smaller
  - compute a pairing in 1.6 ms
    2.5 times faster



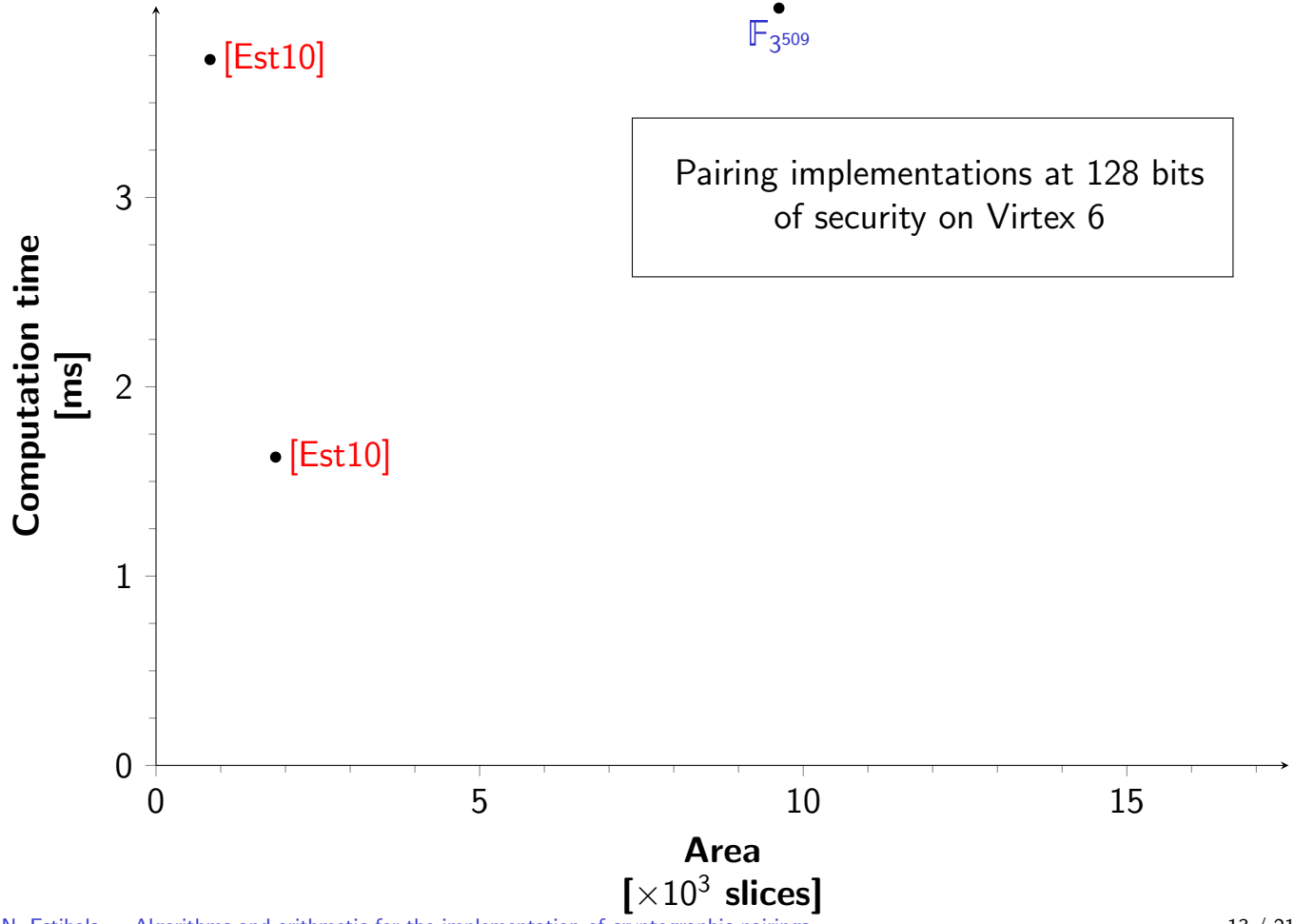
Benchmarks


$\mathbb{F}_{3^{509}}$
Pairing implementations at 128 bits of security on Virtex 6
Computation time [ms]
0
1
2
3
0
5
10
15
Area
[$\times 10^3$ slices]

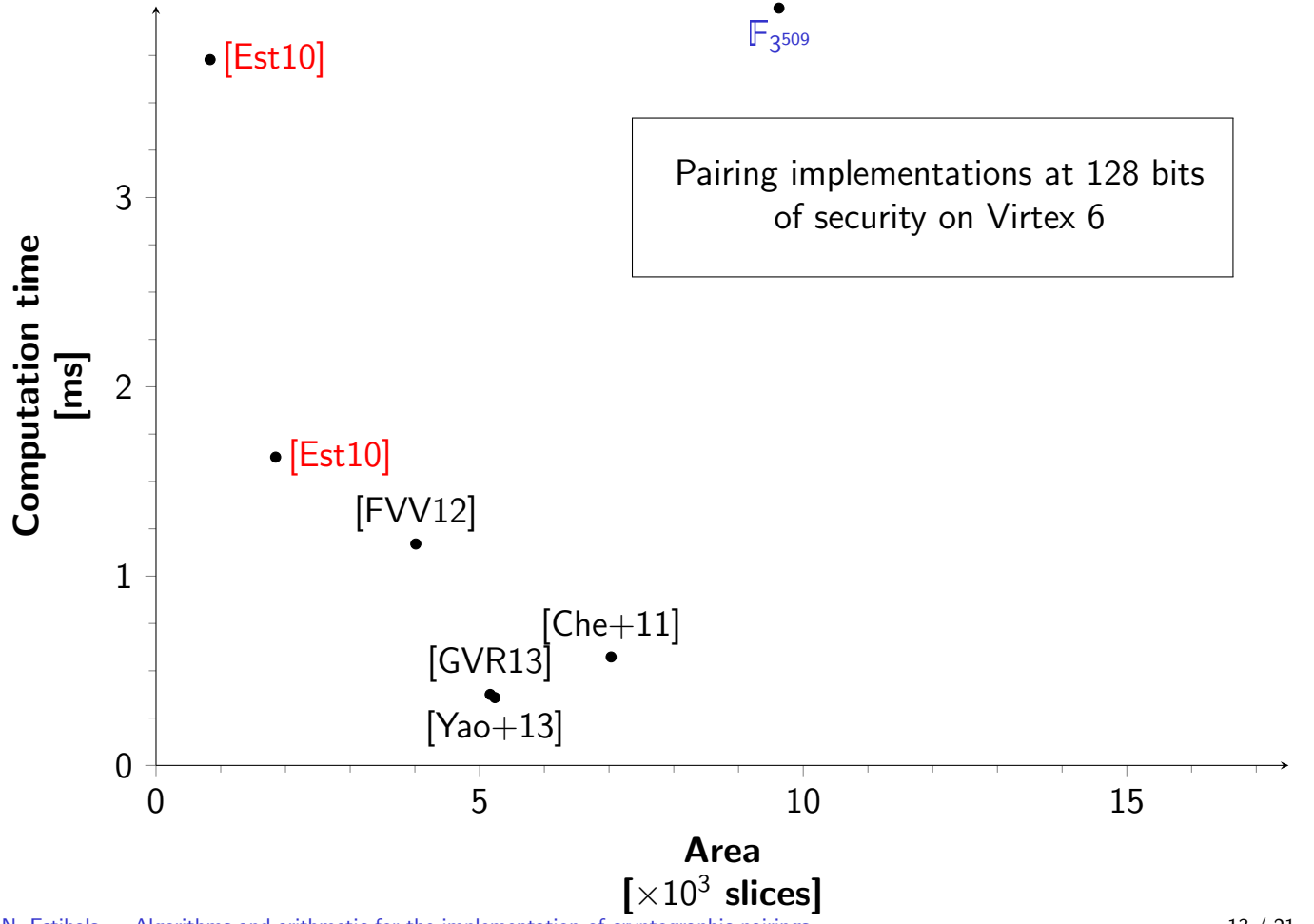
Benchmarks


$\mathbb{F}_{3^{509}}$
Pairing implementations at 128 bits of security on Virtex 6
[Est10]
Computation time [ms]
0
1
2
3
0
5
10
15
Area [$\times 10^3$ slices]

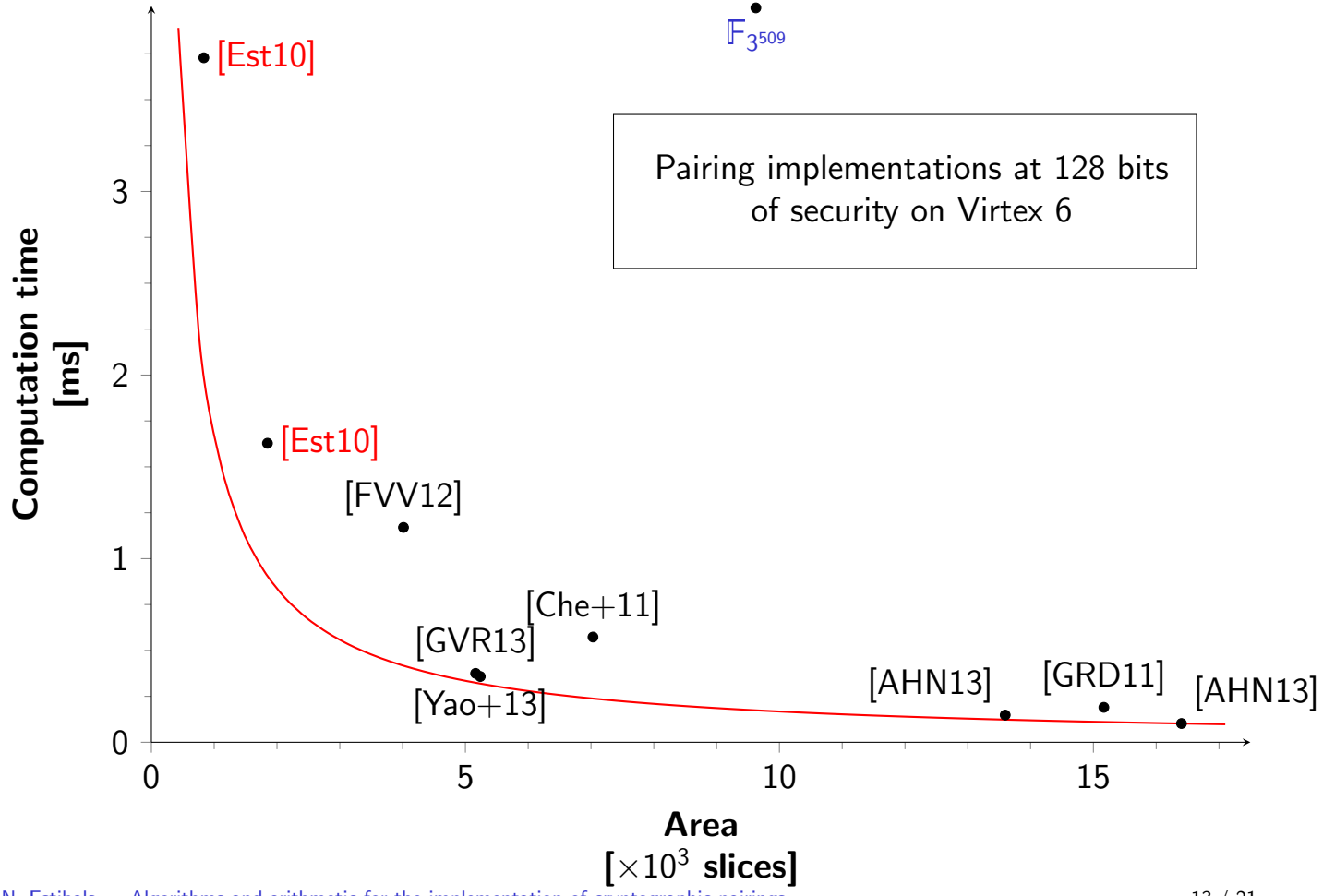
Benchmarks


[Est10]
$\mathbb{F}_{3^{509}}$
Pairing implementations at 128 bits of security on Virtex 6
[Est10]
Computation time [ms]
0
1
2
3
0
5
10
15
Area
[$\times 10^3$ slices]

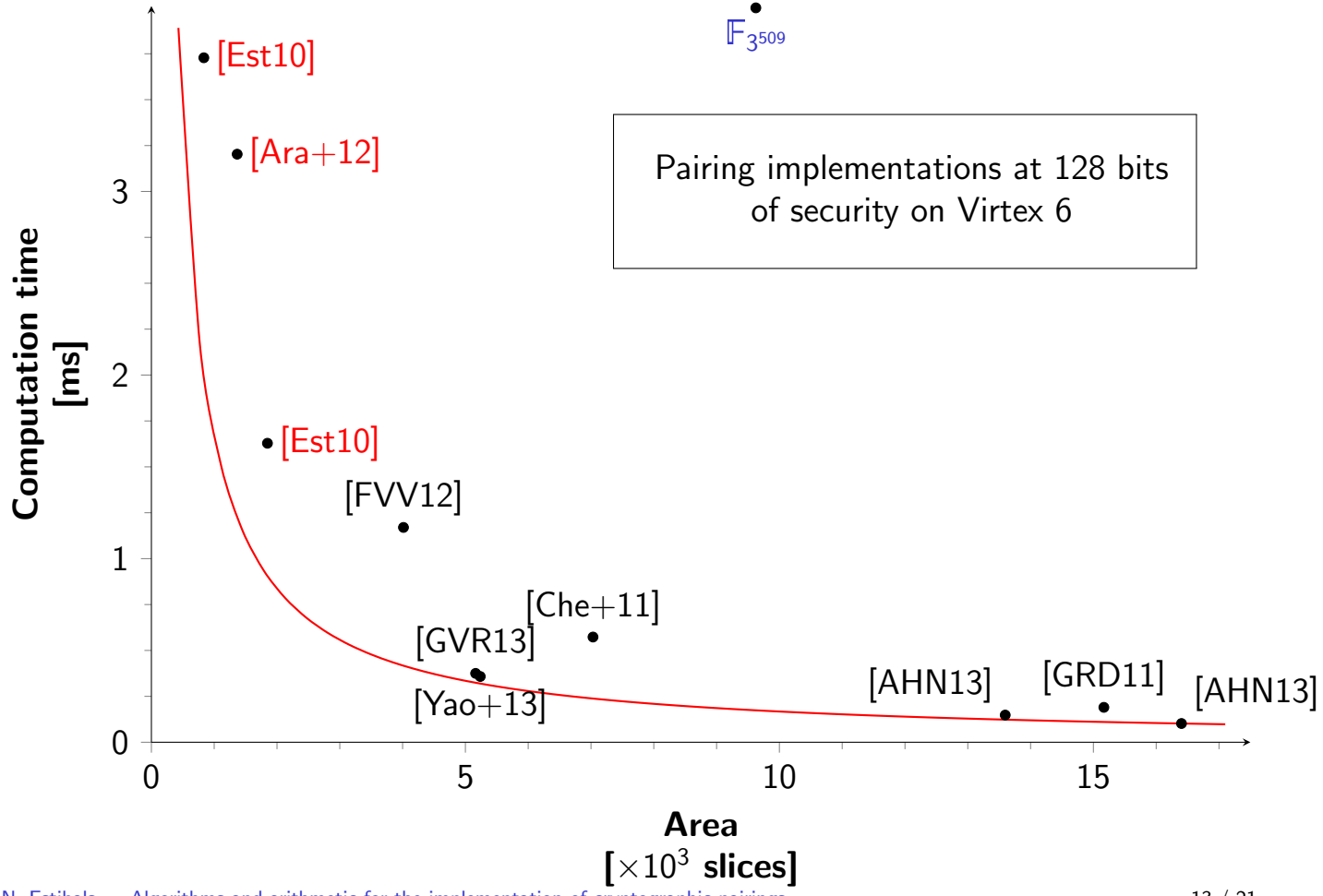
Benchmarks


$\mathbb{F}_{3^{509}}$
[Est10]
Pairing implementations at 128 bits of security on Virtex 6
[Est10]
[FVV12]
[Che+11]
[GVR13]
[Yao+13]
Computation time [ms]
Area [$\times 10^3$ slices]
0
1
2
3
0
5
10
15

Benchmarks


$\mathbb{F}_{3^{509}}$
[Est10]
[Est10]
[FVV12]
[Che+11]
[GVR13]
[Yao+13]
[AHN13]
[GRD11]
[AHN13]
Pairing implementations at 128 bits of security on Virtex 6
Computation time [ms]
Area [$\times 10^3$ slices]
0
1
2
3
0
5
10
15

Benchmarks


$\mathbb{F}_{3^{509}}$
[Est10]
[Ara+12]
[Est10]
[FVV12]
[Che+11]
[GVR13]
[Yao+13]
[AHN13]
[GRD11]
[AHN13]
Pairing implementations at 128 bits of security on Virtex 6
Computation time [ms]
0
1
2
3
0
5
10
15
Area [$\times 10^3$ slices]

Outline of the talk

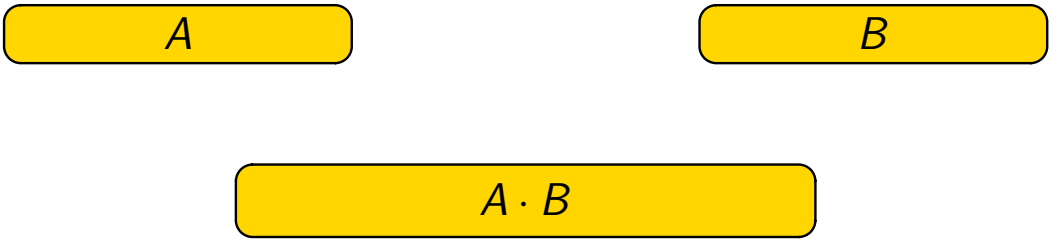
- Compact design through composite extension fields
- Pipelined subquadratic multiplier
- Conclusion and Perspectives

Finite field representation and Karatsuba's formula

- Small characteristic fields
- Polynomial basis representation

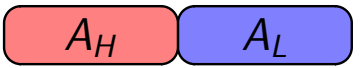
Finite field representation and Karatsuba's formula

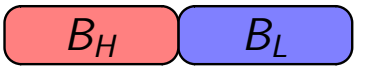
- Small characteristic fields
- Polynomial basis representation
- Multiplication is critical
  - make use of fast arithmetic
  - Karatsuba algorithm for polynomials



Finite field representation and Karatsuba's formula

- Small characteristic fields
- Polynomial basis representation
- Multiplication is critical
  - make use of fast arithmetic
  - Karatsuba algorithm for polynomials

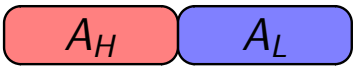


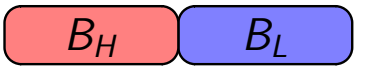


$$A_H B_H X^{2n} + (A_H B_L + A_L B_H) X^n + A_L B_L$$

Finite field representation and Karatsuba's formula

- Small characteristic fields
- Polynomial basis representation
- Multiplication is critical
  - make use of fast arithmetic
  - Karatsuba algorithm for polynomials





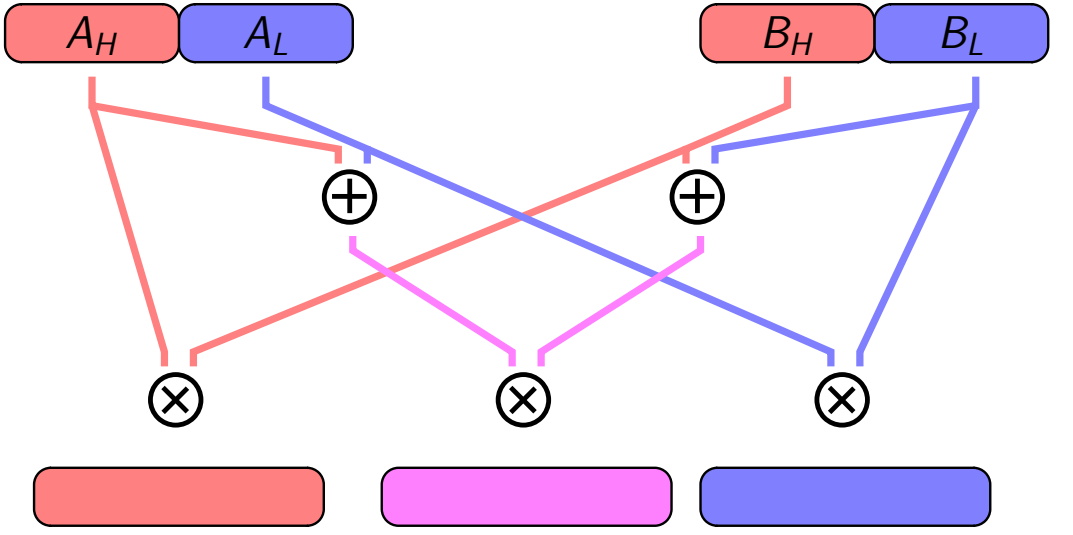
$$A_H B_H X^{2n} + (A_H B_L + A_L B_H) X^n + A_L B_L$$

$$ab' + a'b = (a + a')(b + b') - ab - a'b'$$

$$A_H B_H X^{2n} + ((A_H + A_L)(B_H + B_L) - A_H B_H - A_L B_L) X^n + A_L B_L$$

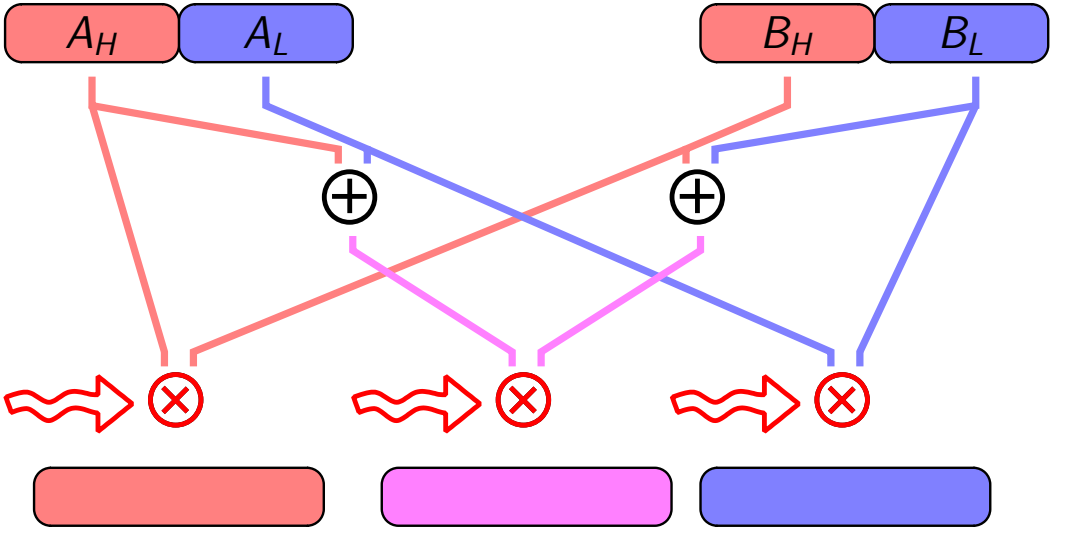
Finite field representation and Karatsuba's formula

- Small characteristic fields
- Polynomial basis representation
- Multiplication is critical
  - make use of fast arithmetic
  - Karatsuba algorithm for polynomials



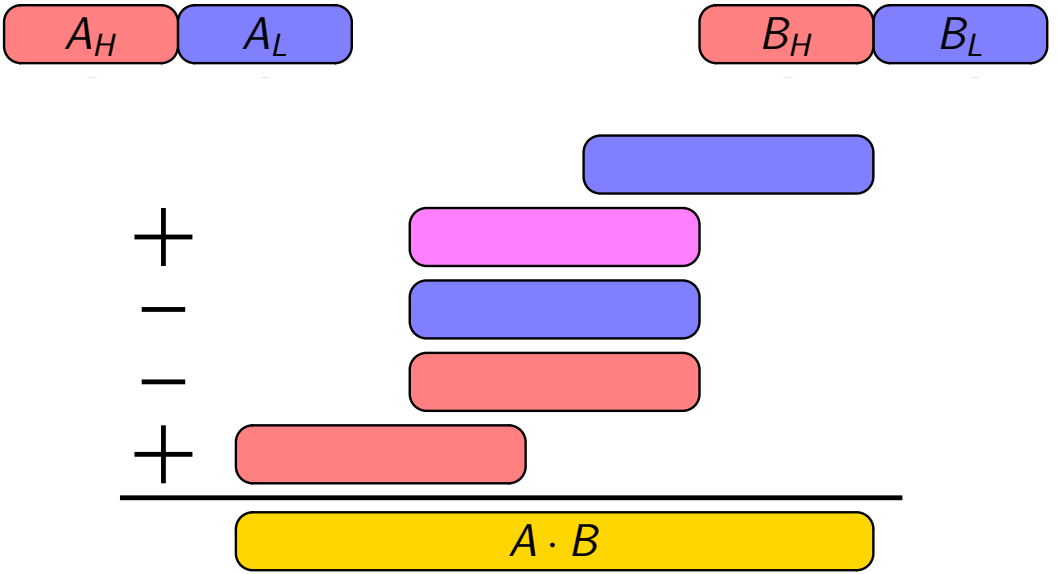
Finite field representation and Karatsuba's formula

- Small characteristic fields
- Polynomial basis representation
- Multiplication is critical
  - make use of fast arithmetic
  - Karatsuba algorithm for polynomials

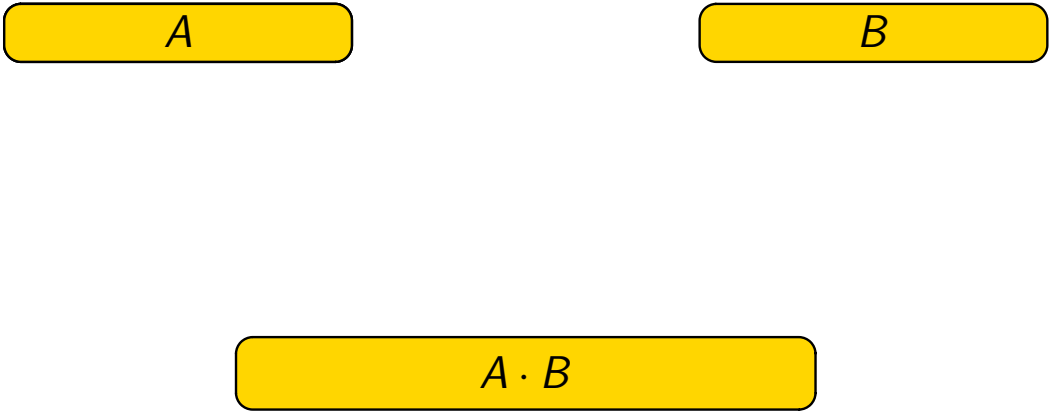


Finite field representation and Karatsuba's formula

- Small characteristic fields
- Polynomial basis representation
- Multiplication is critical
  - make use of fast arithmetic
  - Karatsuba algorithm for polynomials



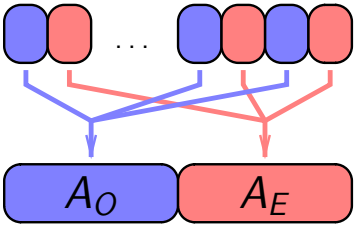
Odd–even split for Karatsuba multiplication

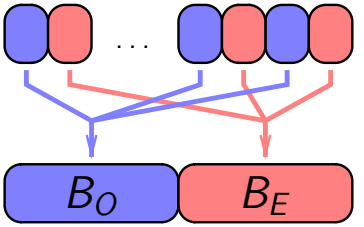


Odd–even split for Karatsuba multiplication

$$(A_O B_O X^2 + A_E B_E) + X(A_O B_E + A_E B_O)$$

Odd–even split for Karatsuba multiplication



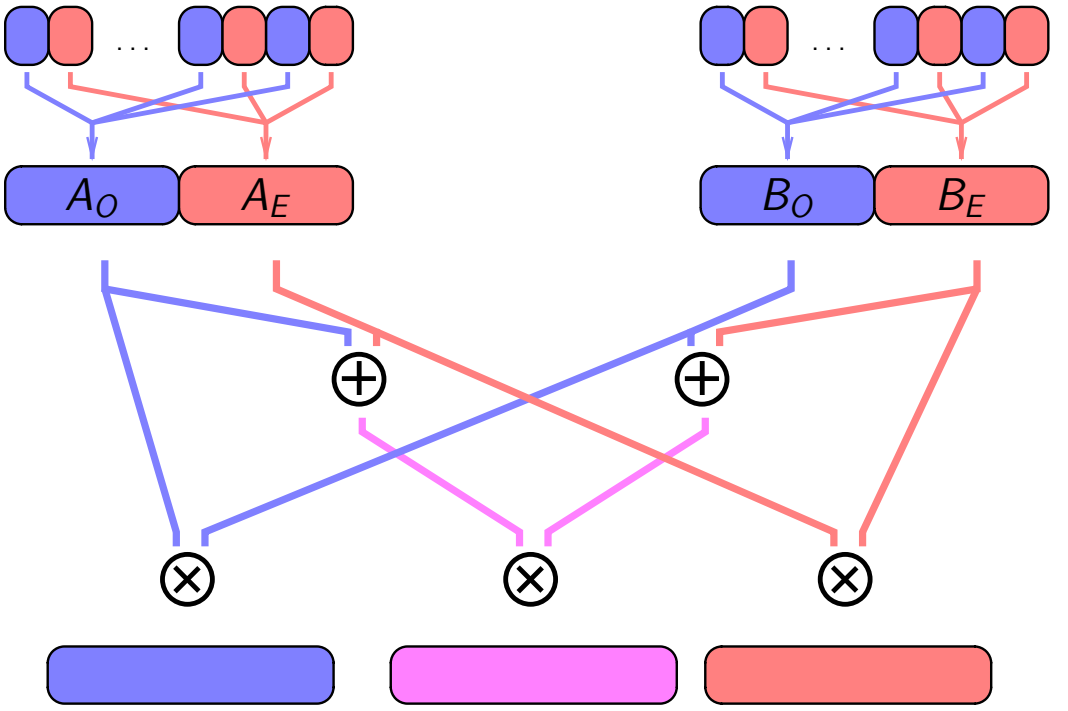


$$(A_O B_O X^2 + A_E B_E) + X(A_O B_E + A_E B_O)$$

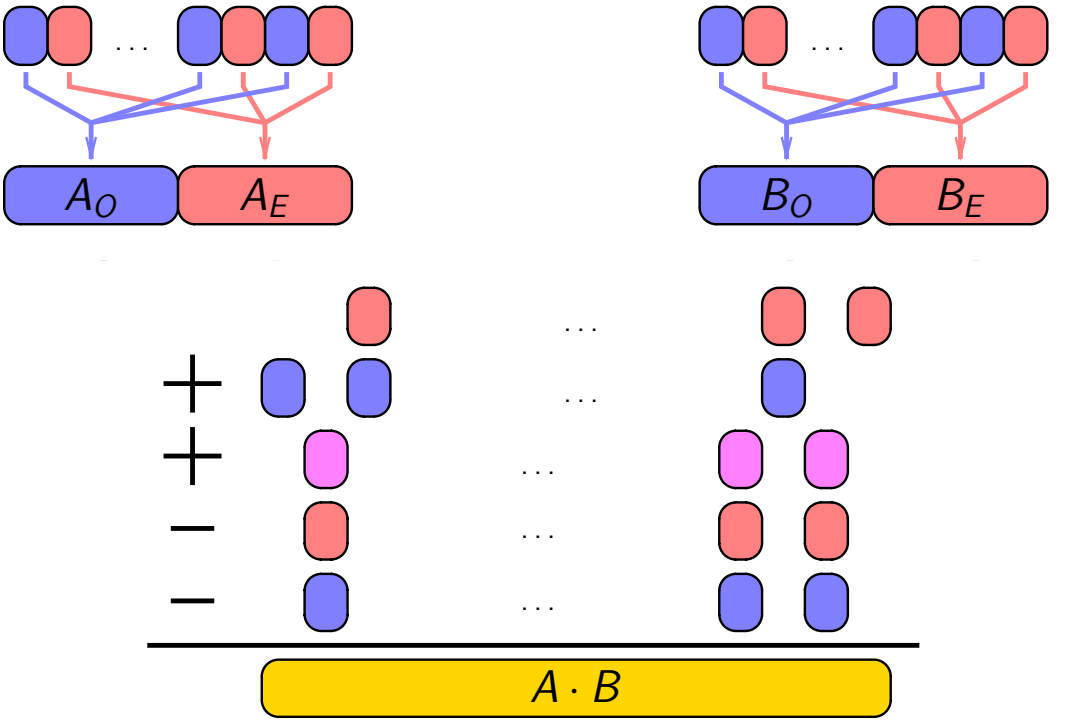
$$ab' + a'b = (a + a')(b + b') - ab - a'b'$$

$$(A_O B_O X^2 + A_E B_E) + X((A_O + A_E)(B_O + B_E) - A_O B_O - A_E B_E)$$

Odd–even split for Karatsuba multiplication

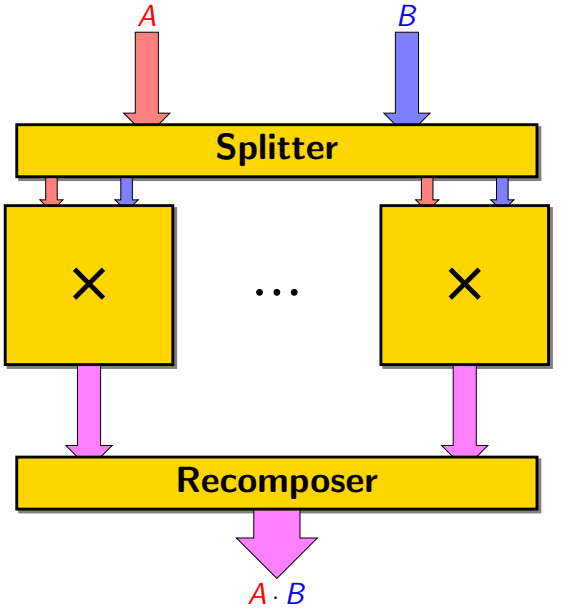


Odd–even split for Karatsuba multiplication



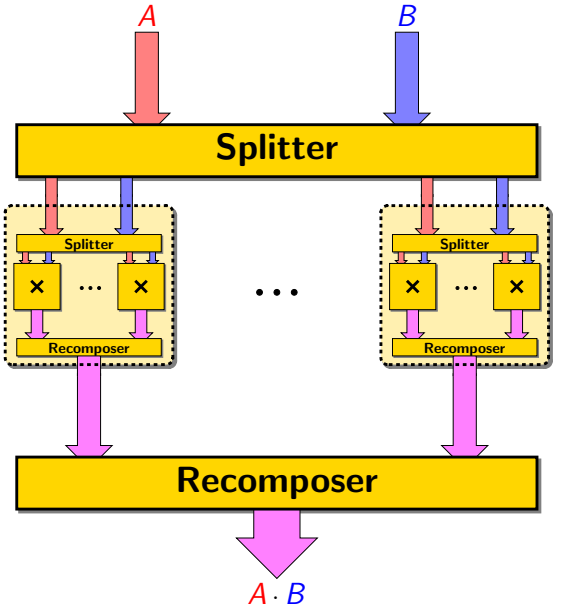
Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts



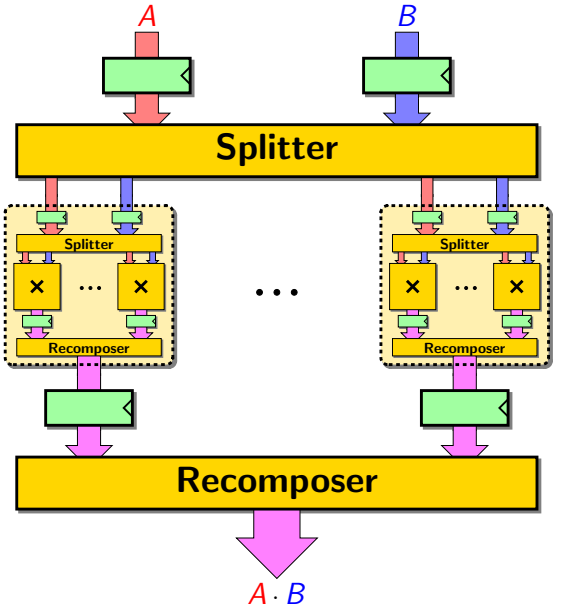
Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts
- Recursive scheme
  - eventually use different multiplication algorithms
  - end with the quadratic paper-and-pencil algorithm



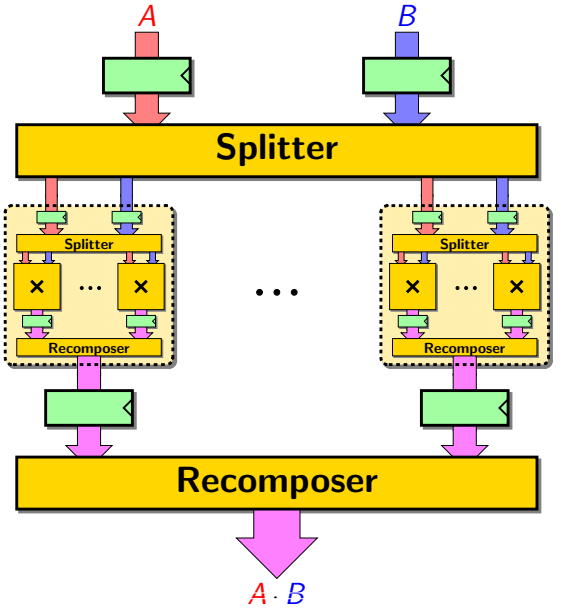
Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts
- Recursive scheme
  - eventually use different multiplication algorithms
  - end with the quadratic paper-and-pencil algorithm
- Pipelined
  - with the help of optional registers
  - cut the critical path
  - increase the frequency



Multiplier architecture

- Karatsuba-like algorithm:
  - split the operands
  - compute the subproducts
  - recompose the result
- Fully parallel evaluation of the subproducts
- Recursive scheme
  - eventually use different multiplication algorithms
  - end with the quadratic paper-and-pencil algorithm
- Pipelined
  - with the help of optional registers
  - cut the critical path
  - increase the frequency
- Generated VHDL code
  - work for all small-characteristic fields
  - compare different recursions
  - configurable pipeline depth



Outline of the talk

- Compact design through composite extension fields
- Pipelined subquadratic multiplier
- Conclusion and Perspectives

Conclusion

- Hardware implementations of pairing

Conclusion

- Hardware implementations of pairing
- General method for cryptographic implementations
  - study mathematical structures
  - fix parameters thanks to cryptanalysis
  - algorithmic optimizations
  - choose the right arithmetic representation
  - implement different hardware accelerators

Perspectives

- Lower-level architecture
  - FPGA is a good prototyping platform
  - but with limited uses in real-life devices
  - develop skills in ASIC designs
  - power consumption awareness

Perspectives

- Lower-level architecture
  - FPGA is a good prototyping platform
  - but with limited uses in real-life devices
  - develop skills in ASIC designs
  - power consumption awareness
- Integrate side-channel counter-measures
  - side-channel attacks are very effective threats
  - embedded systems need to be protected

Perspectives

- Lower-level architecture
  - FPGA is a good prototyping platform
  - but with limited uses in real-life devices
  - develop skills in ASIC designs
  - power consumption awareness
- Integrate side-channel counter-measures
  - side-channel attacks are very effective threats
  - embedded systems need to be protected
- Use this method on different cryptographic primitives
  - scalar multiplication on hyperelliptic curves
  - lattice-based cryptography

Thank you for your attention!

Questions?