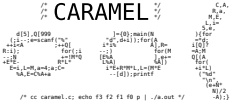
Groupe de travail Arénaire — January 6, 2011

Compact hardware for computing the Tate pairing over 128-bit-security supersingular curves

Nicolas Estibals

CARAMEL project-team, LORIA, Nancy Université / CNRS / INRIA

Nicolas.Estibals@loria.fr



Outline of the talk

- Context
- Pairings-friendly curves with 128 bits of security
- Taking advantage of the tower field
- Performance results
- Conclusion

Pairing-based cryptography

- Origin of pairings in cryptography
  - attack against some elliptic curves
    - ⋆ Menezes–Okamoto–Vanstone, 1993
    - ⋆ Frey–Rück, 1994

Pairing-based cryptography

- Origin of pairings in cryptography
  - attack against some elliptic curves
    - ⋆ Menezes–Okamoto–Vanstone, 1993
    - ⋆ Frey–Rück, 1994
- Constructive properties
  - one-round three-party key exchange
    - ⋆ Joux, 2000
  - short digital signature
    - ⋆ Boneh–Lynn–Shacham, 2001
    - ⋆ Zang–Safavi–Naini–Susilo, 2004
  - identity-based encryption
    - ⋆ Boneh–Franklin, 2001
    - ⋆ Sakai–Kasahara, 2001
  - ...

Pairing-based cryptography

- Origin of pairings in cryptography
  - attack against some elliptic curves
    - ⋆ Menezes–Okamoto–Vanstone, 1993
    - ⋆ Frey–Rück, 1994
- Constructive properties
  - one-round three-party key exchange
    - ⋆ Joux, 2000
  - short digital signature
    - ⋆ Boneh–Lynn–Shacham, 2001
    - ⋆ Zang–Safavi–Naini–Susilo, 2004
  - identity-based encryption
    - ⋆ Boneh–Franklin, 2001
    - ⋆ Sakai–Kasahara, 2001
  - ...
- Standardization in progress
  - ISO/IEC 14888-3
  - IEEE P1363.3

Hardware accelerators for pairing computation

- Pairings are (almost) everywhere!
  - wide range of targets and applications
    - ⋆ low-resource environment (embedded systems, smart card, . . .)
    - ⋆ high-performance computation (bank server, . . .)
  - non-trivial to compute
    - ⋆ complex mathematical structure
    - ⋆ finite field arithmetic
    - ⋆ substantial amount of computation
- Needs in hardware implementation
  - computation not suited to general purpose processor
  - specific targets (*e.g.* smart card)

Hardware accelerators for pairing computation

- Pairings are (almost) everywhere!
  - wide range of targets and applications
    - ⋆ low-resource environment (embedded systems, smart card, . . .)
    - ⋆ high-performance computation (bank server, . . .)
  - non-trivial to compute
    - ⋆ complex mathematical structure
    - ⋆ finite field arithmetic
    - ⋆ substantial amount of computation
- Needs in hardware implementation
  - computation not suited to general purpose processor
  - specific targets (*e.g.* smart card)
- Previous work on FPGA implementations
  - low-security pairings
  - most are performance-oriented designs
- Our goal:
  - AES-128 equivalent security
  - compact accelerator

Tate pairing

- Bilinear pairing:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$$

Tate pairing

- E elliptic curve over $\mathbb{F}_q$
- ℓ large prime dividing $\#E(\mathbb{F}_q)$
  - in general, $\ell \approx \#E(\mathbb{F}_q)$
  - Hasse's bound : $|\#E(\mathbb{F}_q) - (q+1)| \leqslant 2\sqrt{q}$
  - thus, $\ell \approx q$
- $\mathbb{F}_q$-rational ℓ-torsion of E: $E(\mathbb{F}_q)[\ell] = \{P \in E(\mathbb{F}_q) \mid [\ell]P = \mathcal{O}\}$
- Tate pairing:

$$e : E(\mathbb{F}_q)[\ell] \times \mathbb{G}_2 \to \mathbb{G}_T$$

Tate pairing

- E elliptic curve over $\mathbb{F}_q$
- ℓ large prime dividing $\#E(\mathbb{F}_q)$
  - in general, $\ell \approx \#E(\mathbb{F}_q)$
  - Hasse's bound : $|\#E(\mathbb{F}_q) - (q+1)| \leqslant 2\sqrt{q}$
  - thus, $\ell \approx q$
- $\mathbb{F}_q$-rational ℓ-torsion of E: $E(\mathbb{F}_q)[\ell] = \{P \in E(\mathbb{F}_q) \mid [\ell]P = \mathcal{O}\}$
- Embedding degree: k, the smallest integer s. t. $\ell \mid q^k - 1$
- Tate pairing:

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mathbb{G}_T$$

Tate pairing

- E elliptic curve over $\mathbb{F}_q$
- ℓ large prime dividing $\#E(\mathbb{F}_q)$
  - in general, $\ell \approx \#E(\mathbb{F}_q)$
  - Hasse's bound : $|\#E(\mathbb{F}_q) - (q+1)| \leqslant 2\sqrt{q}$
  - thus, $\ell \approx q$
- $\mathbb{F}_q$-rational ℓ-torsion of E: $E(\mathbb{F}_q)[\ell] = \{P \in E(\mathbb{F}_q) \mid [\ell]P = \mathcal{O}\}$
- Embedding degree: k, the smallest integer s. t. $\ell \mid q^k - 1$
- Set of ℓ-th root of unity: $\mu_\ell = \{u \in \mathbb{F}_{q^k}^* \mid u^\ell = 1\}$
- Tate pairing:

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$

Compute the Tate pairing

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \rightarrow \mu_\ell \subset \mathbb{F}_{q^k}^*$$

- Miller's algorithm
- Iterative algorithm
- Complexity:
  - number of iteration proportional to definition field size
  - a multiplication over $\mathbb{F}_{q^k}$ at each iteration

General attacks

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$

- Pollard's ρ on the torsion subgroup $E[\ell]$
  - $\sqrt{\pi\ell/2} \approx \sqrt{\pi q/2}$ group operations
  - complexity exponential in q

General attacks

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$

- Pollard's ρ on the torsion subgroup $E[\ell]$
  - $\sqrt{\pi\ell/2} \approx \sqrt{\pi q/2}$ group operations
  - complexity exponential in q
- Discrete logarithm in finite field multiplicative group $\mathbb{F}_{q^k}^*$
  - FFS or NFS $\to L_{q^k}[1/3, c]$
  - complexity subexponential in q^k

General attacks

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$

- Pollard's ρ on the torsion subgroup $E[\ell]$
  - $\sqrt{\pi\ell/2} \approx \sqrt{\pi q/2}$ group operations
  - complexity exponential in q
- Discrete logarithm in finite field multiplicative group $\mathbb{F}_{q^k}^*$
  - FFS or NFS $\to L_{q^k}[1/3, c]$
  - complexity subexponential in q^k
- k acts as a cursor to balance the complexity of the two attacks
- $k = 12$: optimal for the 128-bit security level

Outline of the talk

- Context
- Pairings-friendly curves with 128 bits of security
- Taking advantage of the tower field
- Performance results
- Conclusion

Supersingular elliptic curves Vs. Barreto–Naehrig curves

- Definitions:

$$E/\mathbb{F}_3: \quad y^2 = x^3 - x + b, b \neq 0$$
$$E/\mathbb{F}_2: \quad y^2 + y = x^3 + x + b$$

- Supersingular curve

- Definition:

$$E/\mathbb{F}_p: \quad y^2 = x^3 + b, b \neq 0,$$
$$p = 36\alpha^4 - 36\alpha^3 + 24\alpha^2 - 6\alpha + 1$$

- Ordinary curve

$\Rightarrow$ Simpler curve arithmetic (efficient tripling formulae)

Supersingular elliptic curves Vs. Barreto–Naehrig curves

- Definitions:

$$E/\mathbb{F}_3: \quad y^2 = x^3 - x + b, b \neq 0$$
$$E/\mathbb{F}_2: \quad y^2 + y = x^3 + x + b$$

- Supersingular curve

$\Rightarrow$ Simpler curve arithmetic (efficient tripling formulae)

- Distortion map, modified pairing:

$$\delta : E(\mathbb{F}_q)[\ell] \rightarrow E(\mathbb{F}_{q^k})[\ell]$$
$$\hat{e}(P, Q) = e(P, \delta(Q))$$

$\Rightarrow$ Symmetric pairing (BN cannot be used with all protocols)

- Definition:

$$E/\mathbb{F}_p: \quad y^2 = x^3 + b, b \neq 0,$$
$$p = 36\alpha^4 - 36\alpha^3 + 24\alpha^2 - 6\alpha + 1$$

- Ordinary curve

- No distortion map

Supersingular elliptic curves Vs. Barreto–Naehrig curves

- Definitions:

$$E/\mathbb{F}_3: \quad y^2 = x^3 - x + b, b \neq 0$$
$$E/\mathbb{F}_2: \quad y^2 + y = x^3 + x + b$$

- Supersingular curve

- Definition:

$$E/\mathbb{F}_p: \quad y^2 = x^3 + b, b \neq 0,$$
$$p = 36\alpha^4 - 36\alpha^3 + 24\alpha^2 - 6\alpha + 1$$

- Ordinary curve

$\Rightarrow$ Simpler curve arithmetic (efficient tripling formulae)

- Distortion map, modified pairing:

$$\delta : E(\mathbb{F}_q)[\ell] \to E(\mathbb{F}_{q^k})[\ell]$$
$$\hat{e}(P, Q) = e(P, \delta(Q))$$

- No distortion map

$\Rightarrow$ Symmetric pairing (BN cannot be used with all protocols)

- Small characteristic field arithmetic

- Modular arithmetic

$\Rightarrow$ No carry, better suited to hardware implementation

Supersingular elliptic curves Vs. Barreto–Naehrig curves

- Definitions:

  $E/\mathbb{F}_3: \quad y^2 = x^3 - x + b, b \neq 0$

  $E/\mathbb{F}_2: \quad y^2 + y = x^3 + x + b$

- Supersingular curve

  $\Rightarrow$ Simpler curve arithmetic (efficient tripling formulae)

- Distortion map, modified pairing:

  $\delta : E(\mathbb{F}_q)[\ell] \rightarrow E(\mathbb{F}_{q^k})[\ell]$

  $\hat{e}(P, Q) = e(P, \delta(Q))$

  $\Rightarrow$ Symmetric pairing (BN cannot be used with all protocols)

- Small characteristic field arithmetic

  $\Rightarrow$ No carry, better suited to hardware implementation

- Small embedding degree ($k = 6$ or 4)

  $\Rightarrow$ Larger field of definition for the same security level. For 128 bits of security:

  $\mathbb{F}_q$ with $q \approx 3^{500}$ or 2^{1150}

- Definition:

  $E/\mathbb{F}_p: \quad y^2 = x^3 + b, b \neq 0,$

  $p = 36\alpha^4 - 36\alpha^3 + 24\alpha^2 - 6\alpha + 1$

- Ordinary curve

- No distortion map

- Modular arithmetic

- Optimal embedding degree ($k = 12$)

  $\mathbb{F}_p$ with p a 256-bit prime.

Supersingular elliptic curves

- Definitions:

$$E/\mathbb{F}_3: \quad y^2 = x^3 - x + b, b \neq 0$$
$$E/\mathbb{F}_2: \quad y^2 + y = x^3 + x + b$$

- Supersingular curve
  $\Rightarrow$ Simpler curve arithmetic (efficient tripling formulae)
- Distortion map, modified pairing:

$$\delta : E(\mathbb{F}_q)[\ell] \rightarrow E(\mathbb{F}_{q^k})[\ell]$$
$$\hat{e}(P, Q) = e(P, \delta(Q))$$
$\Rightarrow$ Symmetric pairing

- Small characteristic field arithmetic
  $\Rightarrow$ No carry, better suited to hardware implementation
- Small embedding degree ($k = 6$ or 4)
  $\Rightarrow$ Larger field of definition for the same security level.
  $\mathbb{F}_q$ with $q \approx 3^{500}$ or 2^{1150}

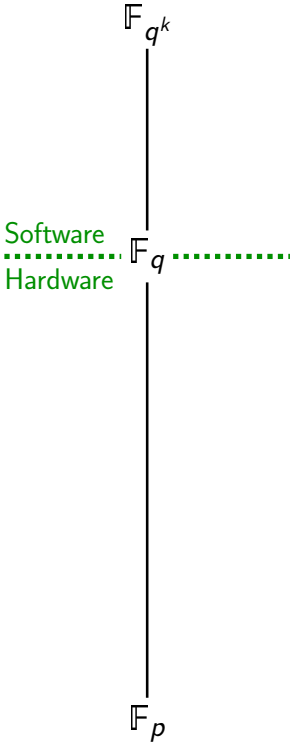
Which field of definition?

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$

- Arithmetic of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$:
  - tower field fixed by pairing construction
  - already optimized by previous works
  - Critical operation: products in $\mathbb{F}_q$

Which field of definition?

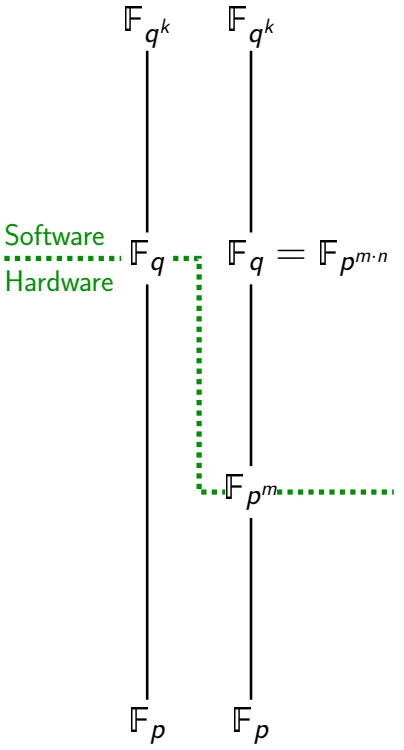
$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$



- Arithmetic of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$:
  - tower field fixed by pairing construction
  - already optimized by previous works
  - Critical operation: products in $\mathbb{F}_q$
- Arithmetic of $\mathbb{F}_q$
  - traditionally implemented in hardware
  - does not scale to the 128-bit security level

Which field of definition?

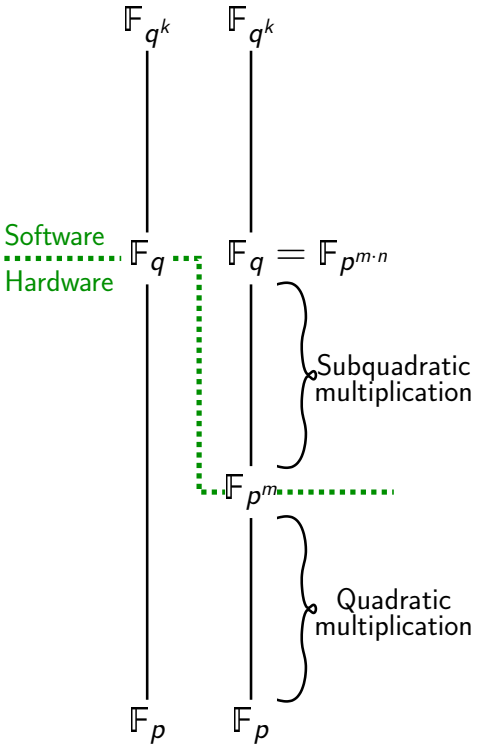
$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$



- Arithmetic of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$:
  - tower field fixed by pairing construction
  - already optimized by previous works
  - Critical operation: products in $\mathbb{F}_q$
- Arithmetic of $\mathbb{F}_q$
  - traditionally implemented in hardware
  - does not scale to the 128-bit security level
- Idea: lower the soft/hardware frontier
  - insert $\mathbb{F}_{p^m}$ in the tower field
  - implement it in hardware

Which field of definition?

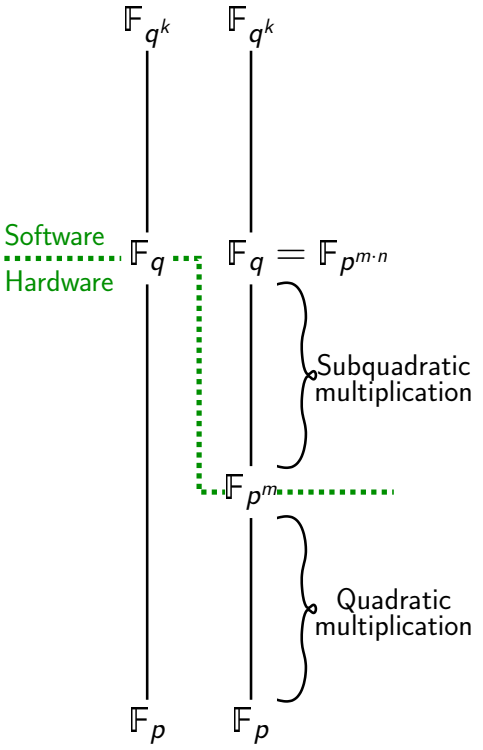
$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$



- Arithmetic of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$:
  - tower field fixed by pairing construction
  - already optimized by previous works
  - Critical operation: products in $\mathbb{F}_q$
- Arithmetic of $\mathbb{F}_q$
  - traditionally implemented in hardware
  - does not scale to the 128-bit security level
- Idea: lower the soft/hardware frontier
  - insert $\mathbb{F}_{p^m}$ in the tower field
  - implement it in hardware
  - use subquadratic multiplication algorithm for $\mathbb{F}_q$ over $\mathbb{F}_{p^m}$

Which field of definition?

$$e : E(\mathbb{F}_q)[\ell] \times E(\mathbb{F}_{q^k})[\ell] \to \mu_\ell \subset \mathbb{F}_{q^k}^*$$



- Arithmetic of $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$:
  - tower field fixed by pairing construction
  - already optimized by previous works
  - Critical operation: products in $\mathbb{F}_q$
- Arithmetic of $\mathbb{F}_q$
  - traditionally implemented in hardware
  - does not scale to the 128-bit security level
- Idea: lower the soft/hardware frontier
  - insert $\mathbb{F}_{p^m}$ in the tower field
  - implement it in hardware
  - use subquadratic multiplication algorithm for $\mathbb{F}_q$ over $\mathbb{F}_{p^m}$
- Problem:
  - field with composite extension degree
  - allows some additional attacks

Weil Descent-based attacks

- We now consider:

$$E(\mathbb{F}_{p^{m \cdot n}})[\ell] \text{ with } m \text{ prime and } n \text{ small}$$

- Weil descent (or Weil restriction to scalar) apply:

$$E(\mathbb{F}_{p^{m \cdot n}}) \cong W_E(\mathbb{F}_{p^m})$$

Weil Descent-based attacks

- We now consider:
$$E(\mathbb{F}_{p^{m \cdot n}})[\ell] \text{ with } m \text{ prime and } n \text{ small}$$
- Weil descent (or Weil restriction to scalar) apply:
$$E(\mathbb{F}_{p^{m \cdot n}}) \cong W_E(\mathbb{F}_{p^m})$$
- Gaudry–Hess–Smart attack:
  - $W_E(\mathbb{F}_{p^m})$ might map to $\mathrm{Jac}(\mathcal{C})$, with $\mathcal{C}$ a curve of genus at least n
  - index calculus algorithm: solve DLP in $\tilde{O}((p^m)^{2-\frac{2}{n}})$

Weil Descent-based attacks

- We now consider:
$$E(\mathbb{F}_{p^{m \cdot n}})[\ell] \text{ with } m \text{ prime and } n \text{ small}$$
- Weil descent (or Weil restriction to scalar) apply:
$$E(\mathbb{F}_{p^{m \cdot n}}) \cong W_E(\mathbb{F}_{p^m})$$
- Gaudry–Hess–Smart attack:
  - $W_E(\mathbb{F}_{p^m})$ might map to $\mathrm{Jac}(\mathcal{C})$, with $\mathcal{C}$ a curve of genus at least n
  - index calculus algorithm: solve DLP in $\tilde{O}((p^m)^{2-\frac{2}{n}})$
- Static Diffie–Hellman problem
  - leakage when reusing private key (*e.g.* ElGamal encryption)
  - Granger's attack: complexity in $\tilde{O}((p^m)^{1-\frac{1}{n+1}})$
  - revoke key after a certain amount of use is an effective workaround

Suitable curves for 128-bit security level

			Cost of the attacks (bits)			
p^m	n	$\log_2 \ell$	Pollard's ρ	FFS		
3^{503}	1	697	342	132		
3^{97}	5	338	163	130		
3^{67}	7	612	300	129		
3^{53}	11	672	330	140		
3^{43}	13	764	376	138		

Suitable curves for 128-bit security level

			Cost of the attacks (bits)			
p^m	n	$\log_2 \ell$	Pollard's ρ	FFS	GHS	SDH
3^{503}	1	697	342	132	–	–
3^{97}	5	338	163	130	245	128
3^{67}	7	612	300	129	182	92
3^{53}	11	672	330	140	152	77
3^{43}	13	764	376	138	125	63

Suitable curves for 128-bit security level

			Cost of the attacks (bits)			
p^m	n	$\log_2 \ell$	Pollard's ρ	FFS	GHS	SDH
3^{503}	1	697	342	**132**	–	–
3^{97}	5	338	163	**130**	245	128
3^{67}	7	612	300	**129**	182	92
3^{53}	11	672	330	**140**	152	77
3^{43}	13	764	376	138	**125**	63

Suitable curves for 128-bit security level

			Cost of the attacks (bits)			
p^m	n	$\log_2 \ell$	Pollard's ρ	FFS	GHS	SDH
3^{503}	1	697	342	**132**	–	–
3^{97}	5	338	163	**130**	245	128
3^{67}	7	612	300	**129**	182	92
3^{53}	11	672	330	**140**	152	77
3^{43}	13	764	376	138	**125**	63

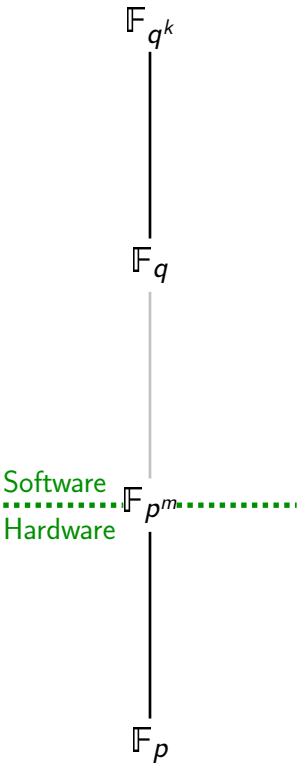
Suitable curves for 128-bit security level

			Cost of the attacks (bits)			
p^m	n	$\log_2 \ell$	Pollard's ρ	FFS	GHS	SDH
3^{503}	1	697	342	**132**	–	–
3^{97}	5	338	163	**130**	245	128
3^{67}	7	612	300	**129**	182	92
3^{53}	11	672	330	**140**	152	77
3^{43}	13	764	376	138	**125**	63
2^{1117}	1	1076	531	**128**	–	–
2^{367}	3	698	342	**128**	489	275
2^{227}	5	733	359	**129**	363	189
2^{163}	7	753	370	**129**	279	142
2^{127}	9	487	236	**130**	225	114
2^{103}	11	922	454	**129**	187	94
2^{89}	13	1044	515	164	**130**	82
2^{73}	15	492	239	136	**127**	68

Outline of the talk

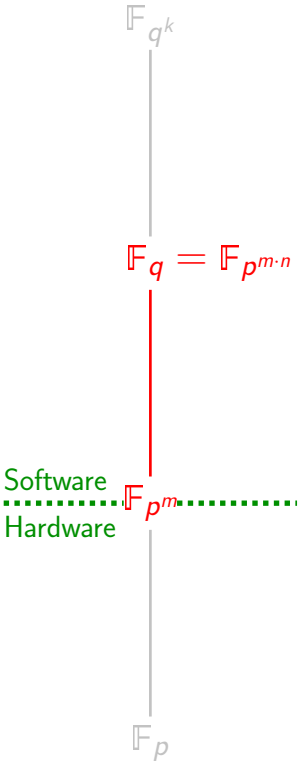
- Context
- Pairings-friendly curves with 128 bits of security
- Taking advantage of the tower field
- Performance results
- Conclusion

Arithmetic of the extension field



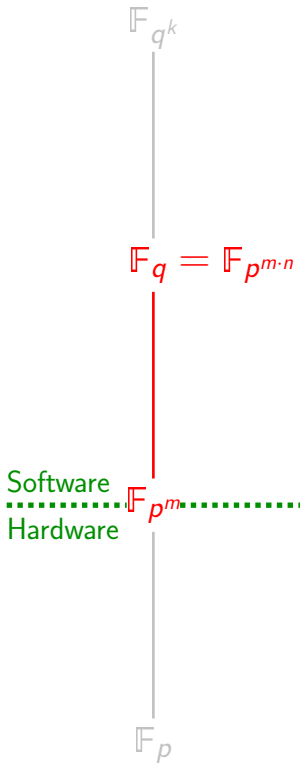
- We already have:
  - pairing algorithm expressed as operations in $\mathbb{F}_q$
  - finite field coprocessor for $\mathbb{F}_{p^m}$

Arithmetic of the extension field



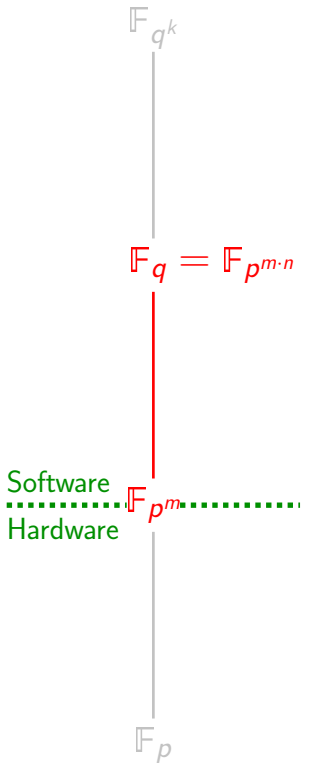
- We already have:
  - pairing algorithm expressed as operations in $\mathbb{F}_q$
  - finite field coprocessor for $\mathbb{F}_{p^m}$
- Polynomial representation: $\mathbb{F}_{p^{m \cdot n}} \cong \mathbb{F}_{p^m}[X]/(f(X))$
  - f irreducible polynomial of degree n

Arithmetic of the extension field



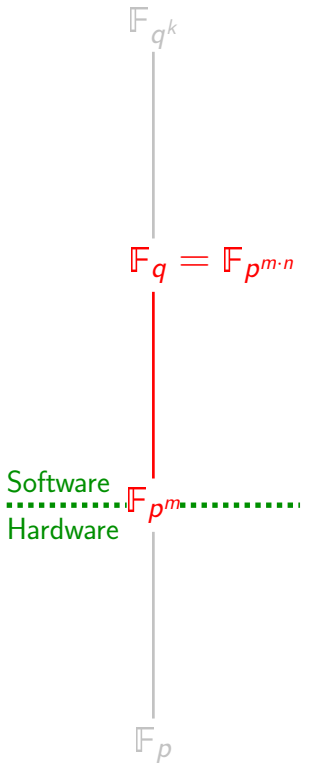
- We already have:
  - pairing algorithm expressed as operations in $\mathbb{F}_q$
  - finite field coprocessor for $\mathbb{F}_{p^m}$
- Polynomial representation: $\mathbb{F}_{p^{m \cdot n}} \cong \mathbb{F}_{p^m}[X]/(f(X))$
  - f irreducible polynomial of degree n
- Addition:
  - add polynomials coefficient-wise
  - no reduction needed

Arithmetic of the extension field



- We already have:
  - pairing algorithm expressed as operations in $\mathbb{F}_q$
  - finite field coprocessor for $\mathbb{F}_{p^m}$
- Polynomial representation: $\mathbb{F}_{p^{m \cdot n}} \cong \mathbb{F}_{p^m}[X]/(f(X))$
  - f irreducible polynomial of degree n
  - f low Hamming weight (tri- or pentanomial)
- Addition:
  - add polynomials coefficient-wise
  - no reduction needed
- Frobenius automorphism: $(.)^p$
  - apply Frobenius on each coefficient
  - linear combination of the coefficients

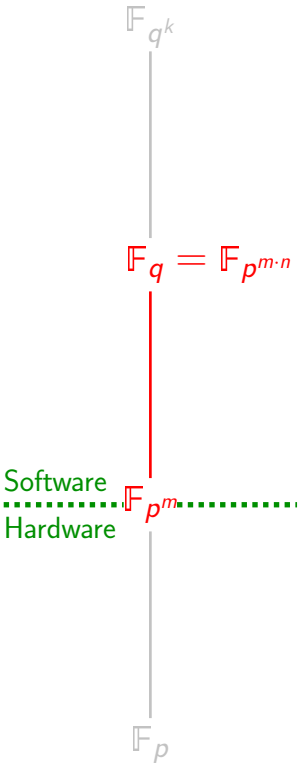
Arithmetic of the extension field



- We already have:
  - pairing algorithm expressed as operations in $\mathbb{F}_q$
  - finite field coprocessor for $\mathbb{F}_{p^m}$
- Polynomial representation: $\mathbb{F}_{p^{m \cdot n}} \cong \mathbb{F}_{p^m}[X]/(f(X))$
  - f irreducible polynomial of degree n
  - f low Hamming weight (tri- or pentanomial)
- Addition:
  - add polynomials coefficient-wise
  - no reduction needed
- Frobenius automorphism: $(.)^p$
  - apply Frobenius on each coefficient
  - linear combination of the coefficients
- Inversion
  - Itoh & Tsujii algorithm (Fermat's little theorem)
  - only need additions, multiplications and Frobenius in $\mathbb{F}_{p^m}$

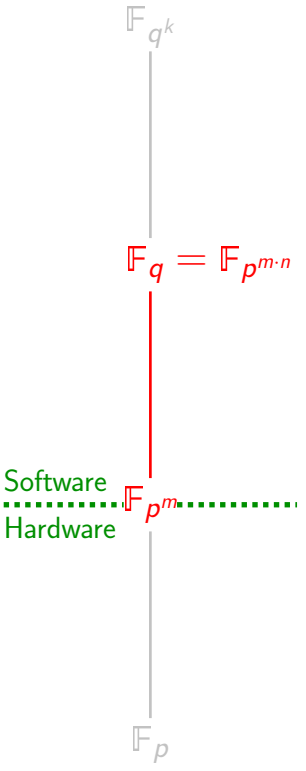
Multiplication in the extension field

- Bottleneck of pairing computation



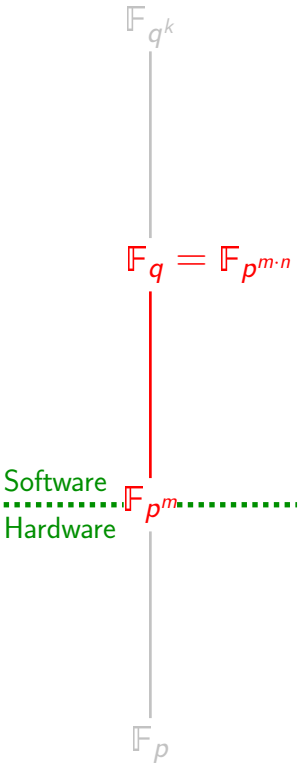
Multiplication in the extension field

- Bottleneck of pairing computation



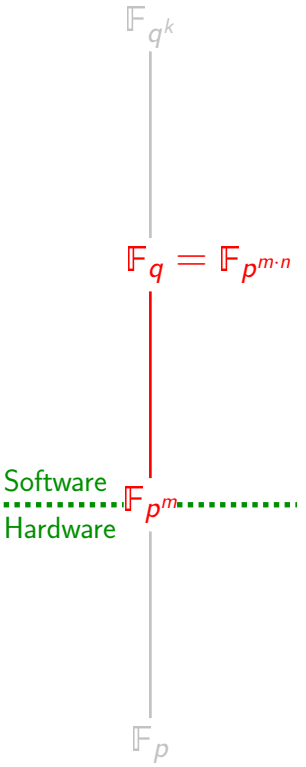
- Our test cases:
  - multiplication in $\mathbb{F}_{3^{97 \cdot 5}}$
    - ⋆ using "schoolbook" algorithm
    - ⋆ 25 products in $\mathbb{F}_{3^{97}}$
    - ⋆ 24 additions in $\mathbb{F}_{3^{97}}$
  - multiplication in $\mathbb{F}_{2^{163 \cdot 7}}$
    - ⋆ using "schoolbook" algorithm
    - ⋆ 49 products in $\mathbb{F}_{2^{163}}$
    - ⋆ 48 additions in $\mathbb{F}_{2^{163}}$

Multiplication in the extension field



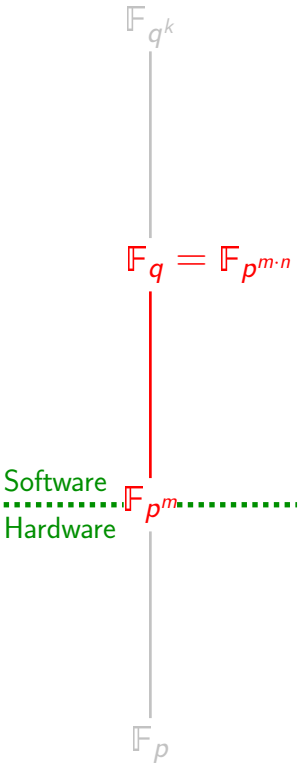
- Bottleneck of pairing computation
- Subquadratic multiplication algorithm
  - Karatsuba
  - Karatsuba with Montgomery's trick
  - Montgomery's formulae
  - CRT-based algorithms
- Our test cases:
  - multiplication in $\mathbb{F}_{3^{97 \cdot 5}}$
    - ⋆ using "schoolbook" algorithm
    - ⋆ 25 products in $\mathbb{F}_{3^{97}}$
    - ⋆ 24 additions in $\mathbb{F}_{3^{97}}$
  - multiplication in $\mathbb{F}_{2^{163 \cdot 7}}$
    - ⋆ using "schoolbook" algorithm
    - ⋆ 49 products in $\mathbb{F}_{2^{163}}$
    - ⋆ 48 additions in $\mathbb{F}_{2^{163}}$

Multiplication in the extension field



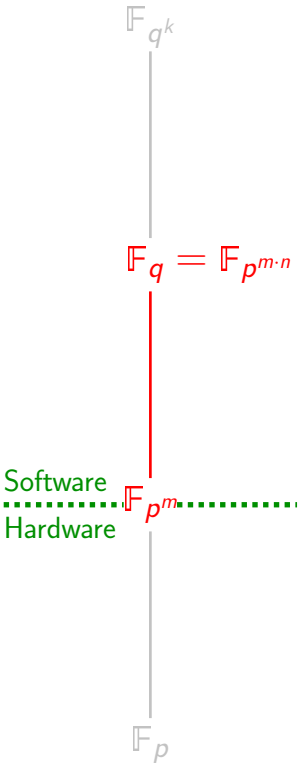
- Bottleneck of pairing computation
- Subquadratic multiplication algorithm
  - Karatsuba
  - Karatsuba with Montgomery's trick
  - Montgomery's formulae
  - CRT-based algorithms
- Our test cases:
  - multiplication in $\mathbb{F}_{3^{97 \cdot 5}}$
    - ⋆ using CRT-based algorithm
    - ⋆ 12 products in $\mathbb{F}_{3^{97}}$
    - ⋆ 53 additions in $\mathbb{F}_{3^{97}}$
  - multiplication in $\mathbb{F}_{2^{163 \cdot 7}}$
    - ⋆ using "schoolbook" algorithm
    - ⋆ 49 products in $\mathbb{F}_{2^{163}}$
    - ⋆ 48 additions in $\mathbb{F}_{2^{163}}$

Multiplication in the extension field



- Bottleneck of pairing computation
- Subquadratic multiplication algorithm
  - Karatsuba
  - Karatsuba with Montgomery's trick
  - Montgomery's formulae
  - CRT-based algorithms
- Our test cases:
  - multiplication in $\mathbb{F}_{3^{97 \cdot 5}}$
    - ⋆ using CRT-based algorithm
    - ⋆ 12 products in $\mathbb{F}_{3^{97}}$
    - ⋆ 53 additions in $\mathbb{F}_{3^{97}}$
  - multiplication in $\mathbb{F}_{2^{163 \cdot 7}}$
    - ⋆ using Montgomery's formulae
    - ⋆ 22 products in $\mathbb{F}_{2^{163}}$
    - ⋆ 84 additions in $\mathbb{F}_{2^{163}}$

Multiplication in the extension field



- Bottleneck of pairing computation
- Subquadratic multiplication algorithm
  - Karatsuba
  - Karatsuba with Montgomery's trick
  - Montgomery's formulae
  - CRT-based algorithms
- Our test cases:
  - multiplication in $\mathbb{F}_{3^{97 \cdot 5}}$
    - ⋆ using CRT-based algorithm
    - ⋆ 12 products in $\mathbb{F}_{3^{97}}$
    - ⋆ 53 additions in $\mathbb{F}_{3^{97}}$
  - multiplication in $\mathbb{F}_{2^{163 \cdot 7}}$
    - ⋆ using Montgomery's formulae
    - ⋆ 22 products in $\mathbb{F}_{2^{163}}$
    - ⋆ 84 additions in $\mathbb{F}_{2^{163}}$
- Choice of algorithm depends on relative cost of multiplication and addition

Count of operations

- Full pairing computation over $E(\mathbb{F}_{3^{97\cdot 5}})$

	$\times$	$+$	$(.)^3$	$1/.$
$\mathbb{F}_{3^{97\cdot 5}}$	3104	13127	4123	1

Count of operations

- Full pairing computation over $E(\mathbb{F}_{3^{97\cdot 5}})$

	$\times$	$+$	$(.)^3$	$1/.$
$\mathbb{F}_{3^{97\cdot 5}}$	3104	13127	4123	1
$\mathbb{F}_{3^{97}}$	37248	253185	20615	1

Count of operations

- Full pairing computation over $E(\mathbb{F}_{3^{97\cdot 5}})$

	$\times$	$+$	$(.)^3$	$1/.$
$\mathbb{F}_{3^{97\cdot 5}}$	3104	13127	4123	1
$\mathbb{F}_{3^{97}}$	37289	253314	21099	–

Count of operations

- Full pairing computation over $E(\mathbb{F}_{3^{97\cdot 5}})$

	$\times$	$+$	$(.)^3$	$1/.$
$\mathbb{F}_{3^{97\cdot 5}}$	3104	13127	4123	1
$\mathbb{F}_{3^{97}}$	37289	253314	21099	–

- Full pairing computation over $E(\mathbb{F}_{2^{163\cdot 7}})$

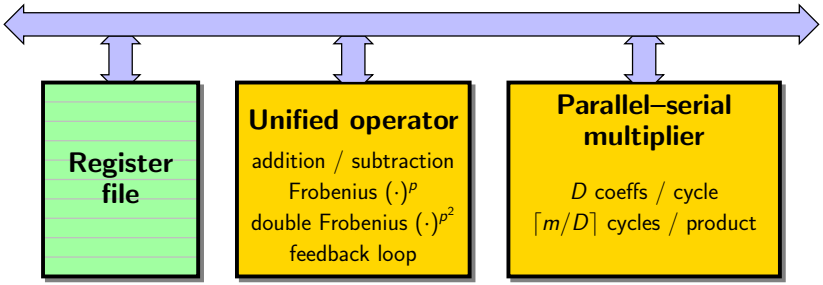
	$\times$	$+$	$(.)^2$	$1/.$
$\mathbb{F}_{2^{163\cdot 7}}$	4019	12605	7424	1
$\mathbb{F}_{2^{163}}$	88509	448361	52782	–

Outline of the talk

- Context
- Pairings-friendly curves with 128 bits of security
- Taking advantage of the tower field
- Performance results
- Conclusion

Experimental setup

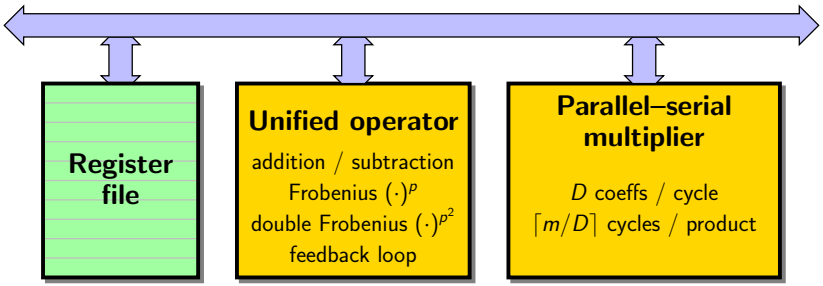
- Finite field coprocessors



  - Prototyped on Xilinx Virtex-4 LX FPGAs
  - Post-place-and-route timing and area estimations

Experimental setup

- Finite field coprocessors

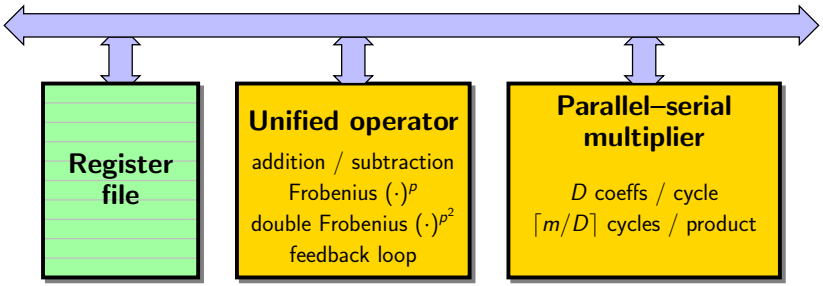


  - Prototyped on Xilinx Virtex-4 LX FPGAs
  - Post-place-and-route timing and area estimations

- Scheduled operations for full pairing computation
  - on $E(\mathbb{F}_{3^{97 \cdot 5}})$, 428,854 cycles
  - on $E(\mathbb{F}_{2^{163 \cdot 7}})$, 1,147,131 cycles

Experimental setup

- Finite field coprocessors

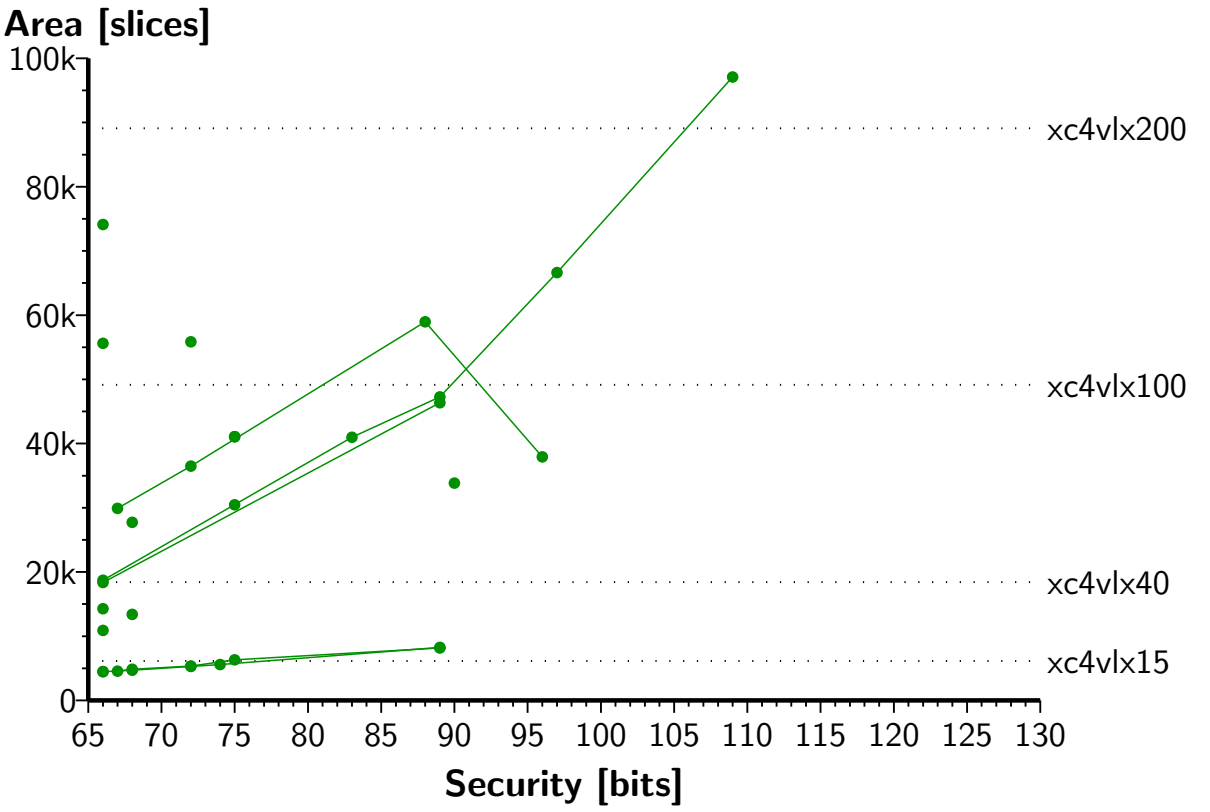


  - Prototyped on Xilinx Virtex-4 LX FPGAs
  - Post-place-and-route timing and area estimations

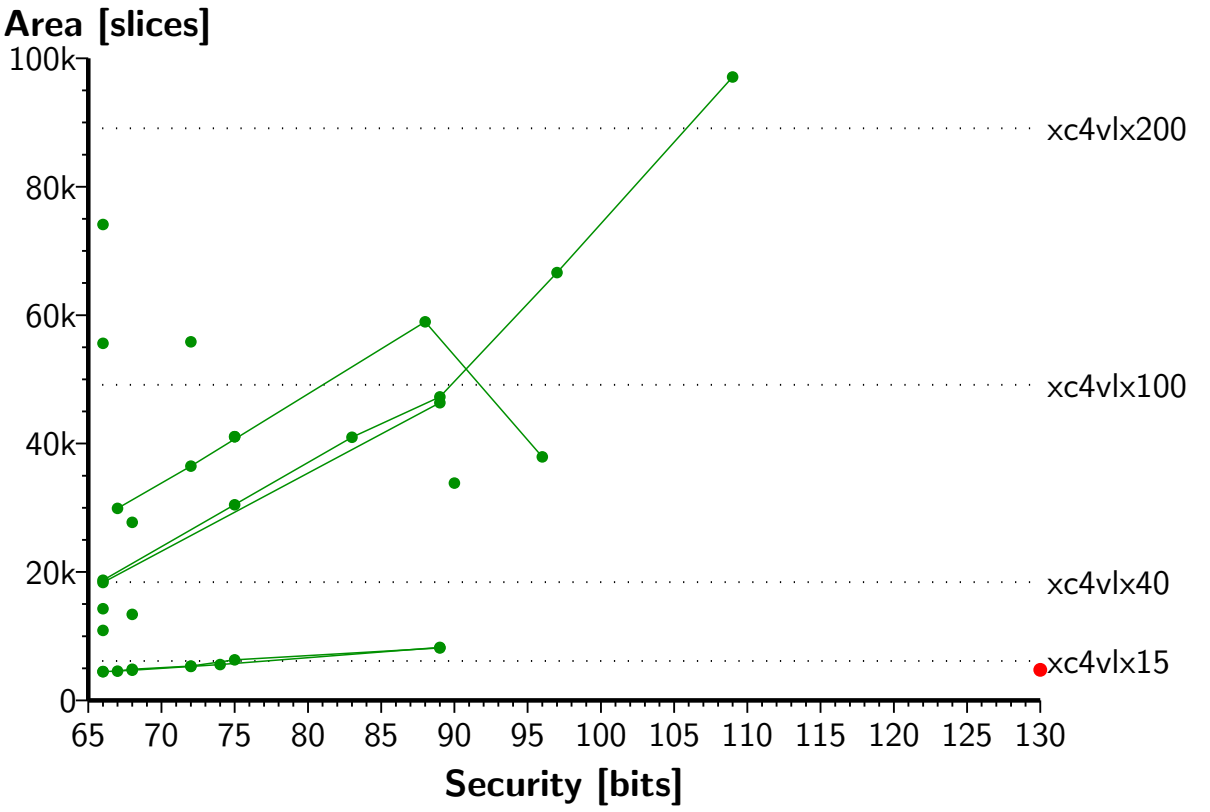
- Scheduled operations for full pairing computation
  - on $E(\mathbb{F}_{3^{97 \cdot 5}})$, 428,854 cycles
  - on $E(\mathbb{F}_{2^{163 \cdot 7}})$, 1,147,131 cycles

- Need of a code factorization method
  - arithmetic of $\mathbb{F}_{p^{m \cdot n}}$ is microcoded

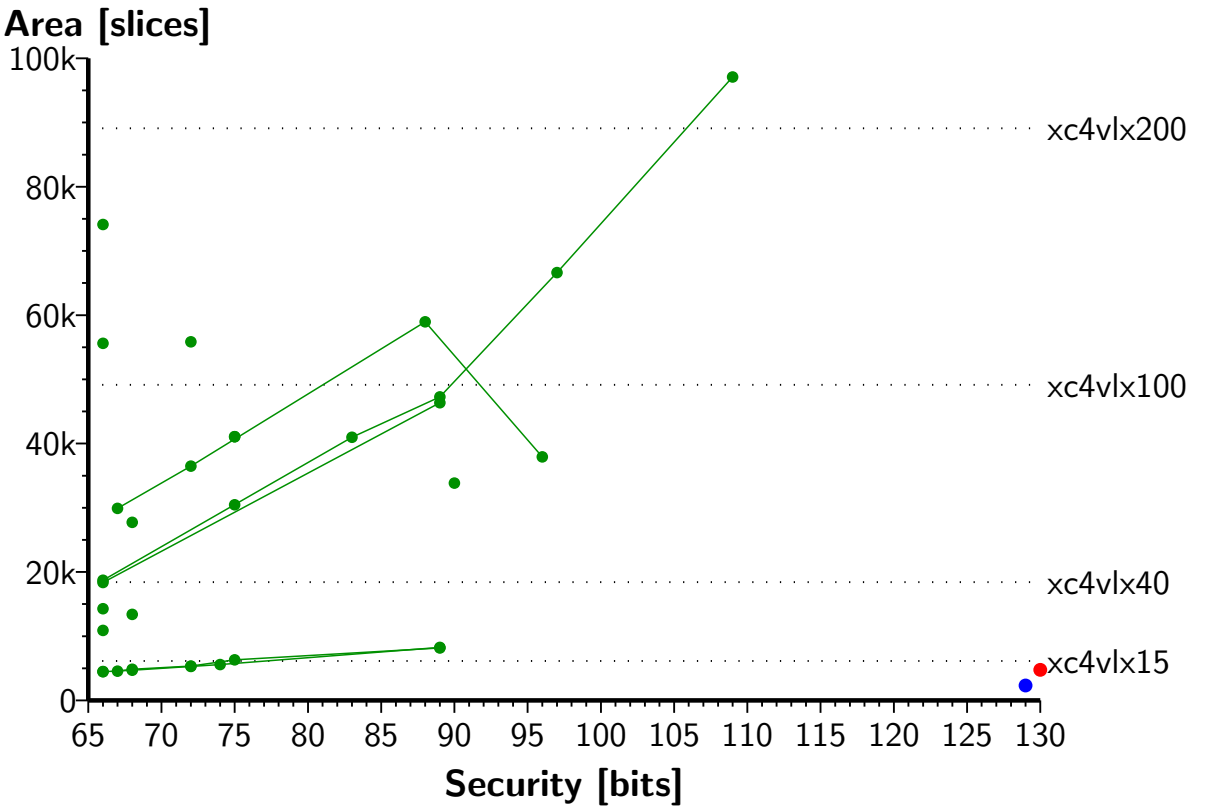
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc4vlx100
xc4vlx40
xc4vlx15
65
70
75
80
85
90
95
100
105
110
115
120
125
130
Security [bits]

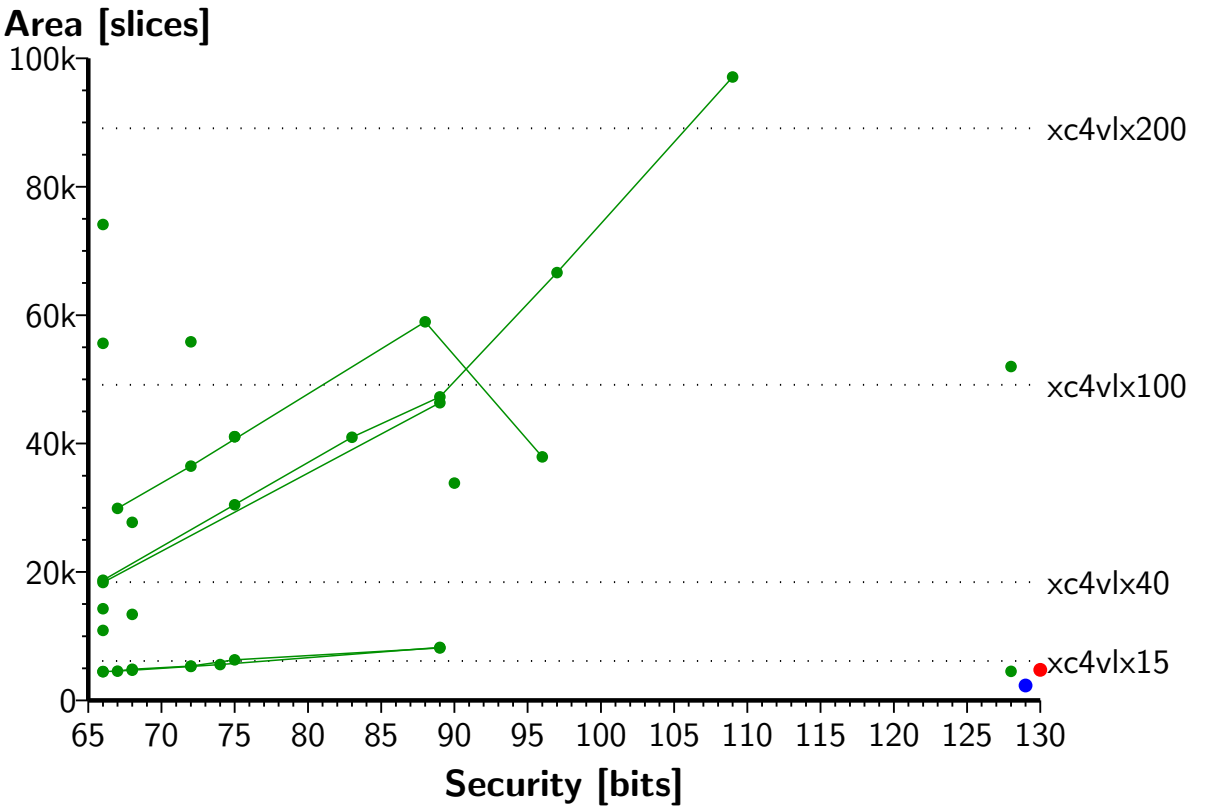
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc4vlx100
xc4vlx40
xc4vlx15
65
70
75
80
85
90
95
100
105
110
115
120
125
130
Security [bits]

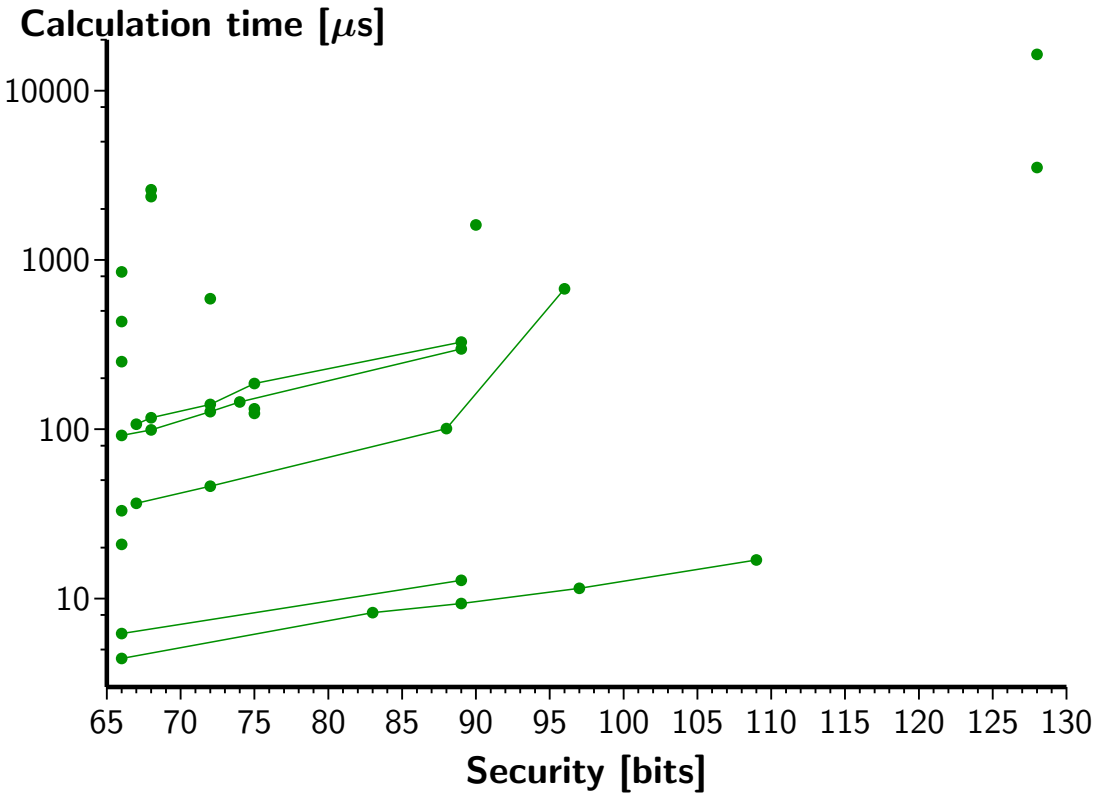
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc4vlx100
xc4vlx40
xc4vlx15
65
70
75
80
85
90
95
100
105
110
115
120
125
130
Security [bits]

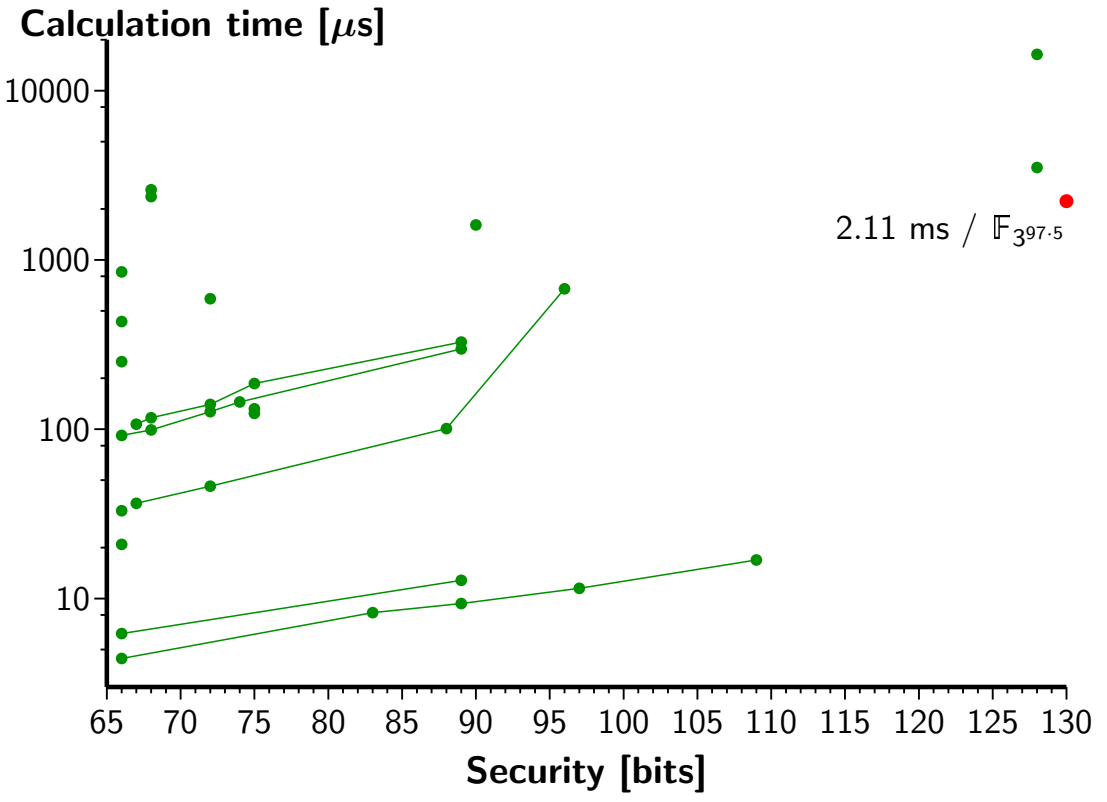
Area


Area [slices]
100k
80k
60k
40k
20k
0
xc4vlx200
xc4vlx100
xc4vlx40
xc4vlx15
65
70
75
80
85
90
95
100
105
110
115
120
125
130
Security [bits]

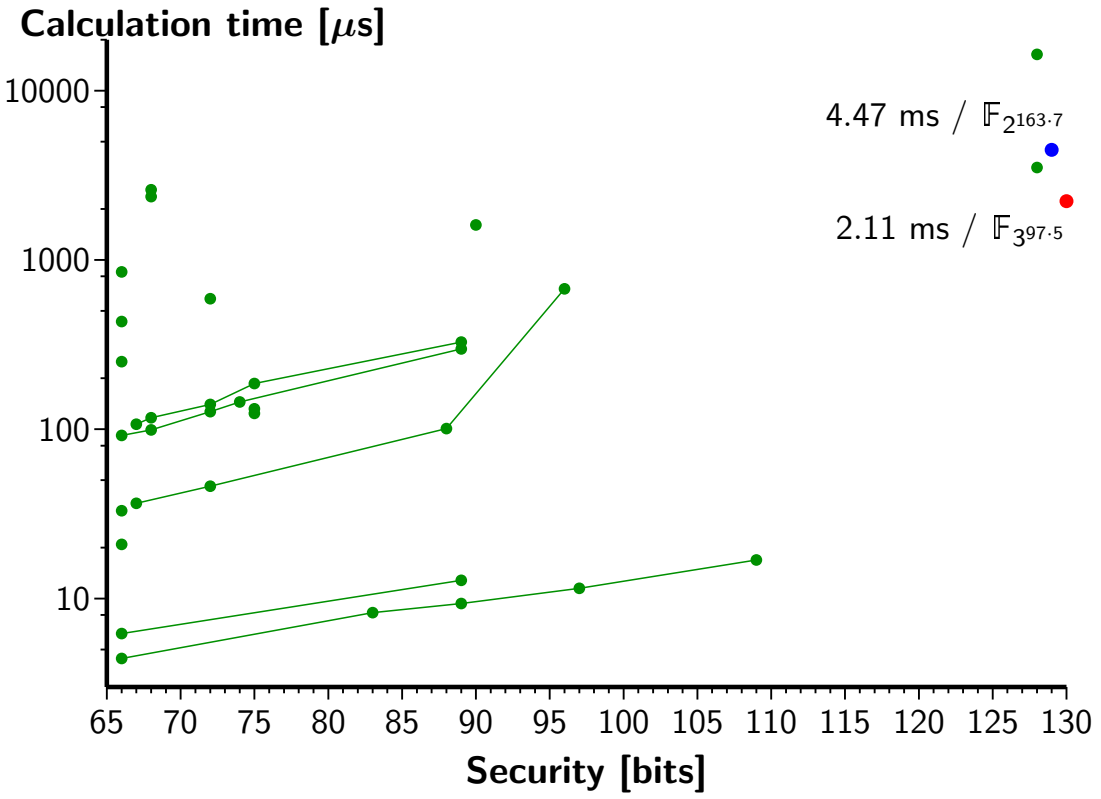
Calculation time



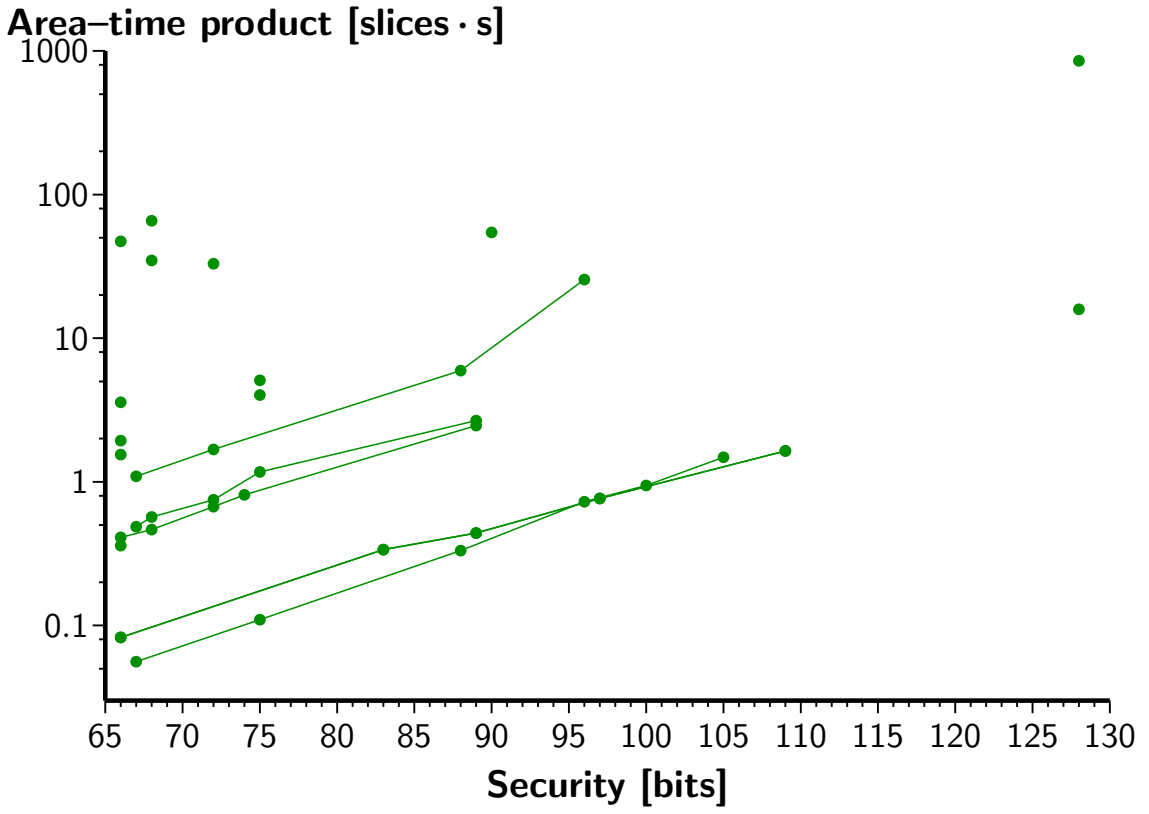
Calculation time



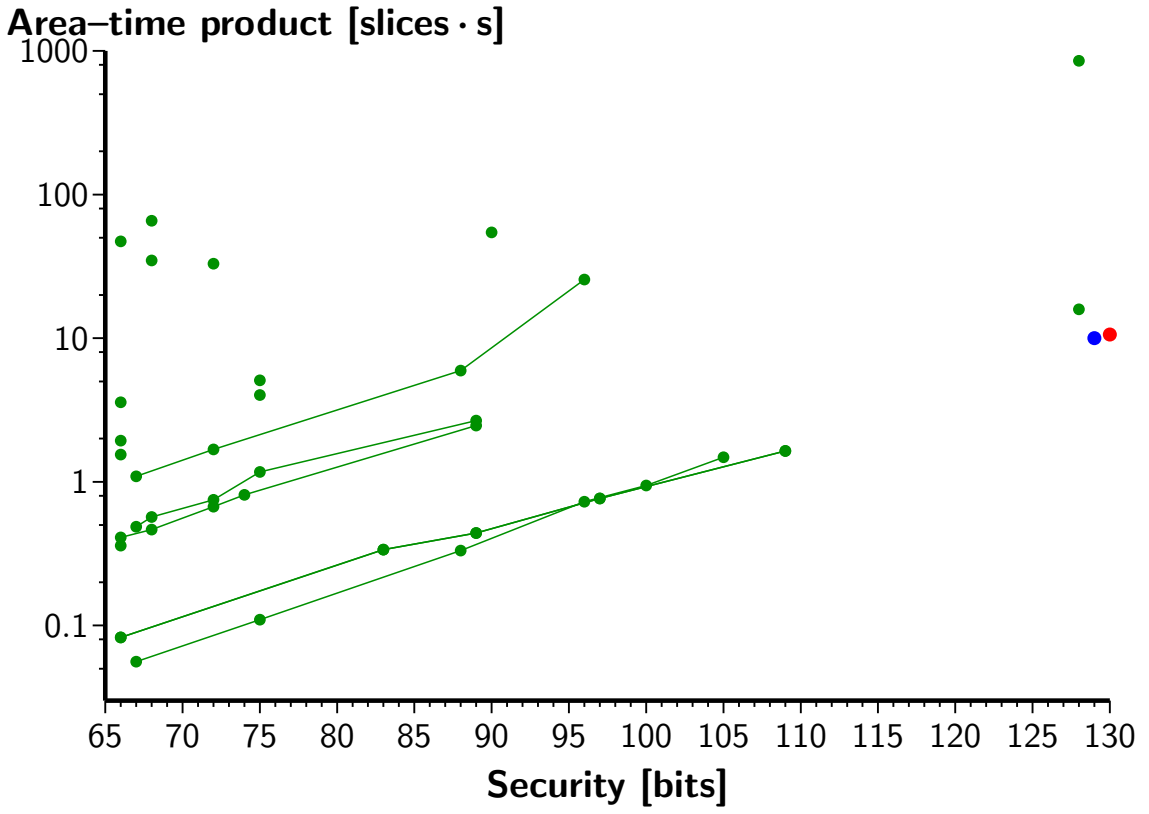
Calculation time



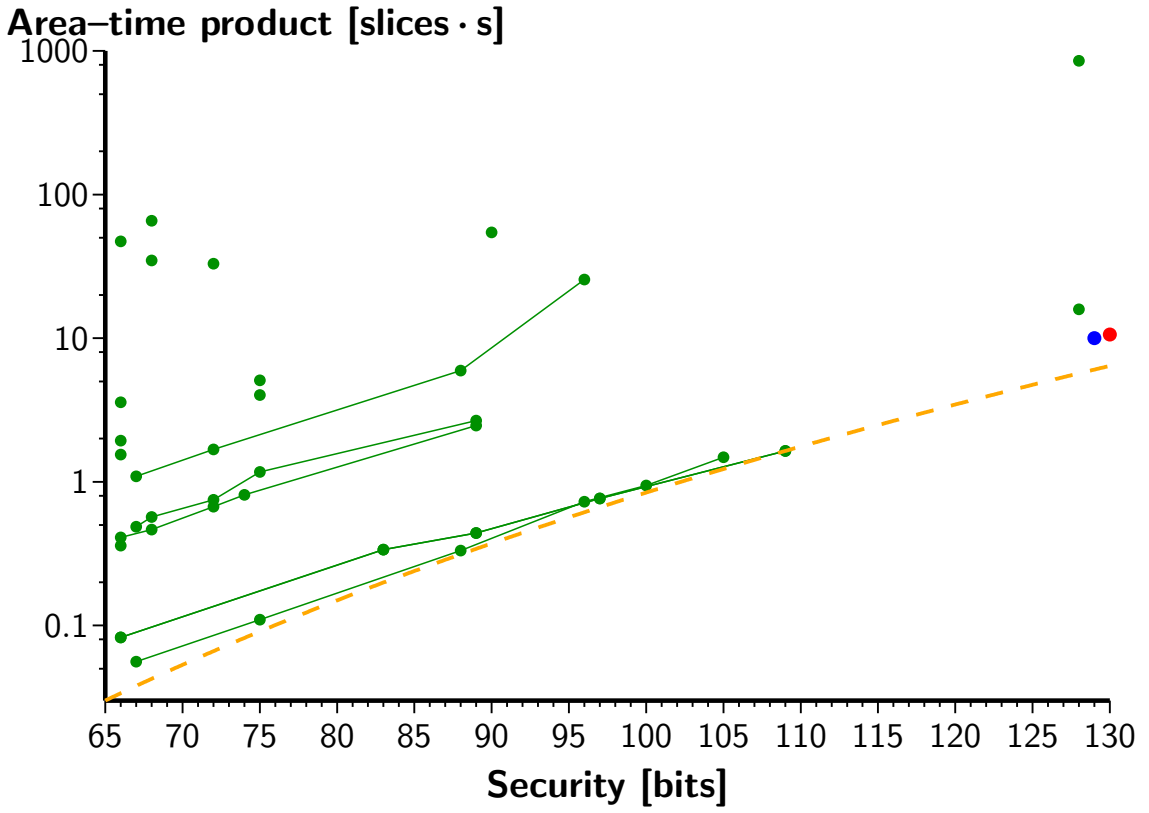
Area–Time product



Area–Time product



Area–Time product



Comparison with ASIC and software

	Supersingular curves	BN-curves
FPGA	2.11 ms (This Work)	52 ms (Ghosh *et al.*, 2010)
ASIC	–	2.91 ms (Fan *et al.*, 2009)
Software (2.4 GHz Intel Core2)	7.59 ms (Beuchat *et al.*, 2009)	0.92 ms (Aranha *et al.*, 2010)

Outline of the talk

- Context
- Pairings-friendly curves with 128 bits of security
- Taking advantage of the tower field
- Performance results
- Conclusion

Conclusion

- Compact, yet reasonably fast, accelerator for pairings with 128 bits of security
  - supersingular elliptic curve
  - low characteristic
  - take advantage of the sub-optimal k to implement efficient field arithmetic

Conclusion

- Compact, yet reasonably fast, accelerator for pairings with 128 bits of security
  - supersingular elliptic curve
  - low characteristic
  - take advantage of the sub-optimal k to implement efficient field arithmetic
- Implement this pairing on more curves:
  - better understanding of the software/hardware frontier
  - hopefully improve performance
  - try higher security level
  - study genus-2 supersingular curves

Thank you for you attention!

Questions?