ECC 2010 — Redmond, USA

Faster Implementation of Pairings

Francisco Rodríguez-Henríquez
CINVESTAV, IPN, Mexico City, Mexico

Joint work with:

Jean-Luc Beuchat	LCIS, University of Tsukuba, Japan
Nicolas Brisebarre	Arénaire, LIP, ÉNS Lyon, France
Jérémie Detrey	Caramel, INRIA Nancy Grand-Est, France
Nicolas Estibals	Caramel, INRIA Nancy Grand-Est, France
Jorge González-Díaz	CINVESTAV, IPN, Mexico City, Mexico
Emmanuel López-Trejo	Intel Guadalajara Design Center, Mexico
Luis Martínez-Ramos	CINVESTAV, IPN, Mexico City, Mexico
Shigeo Mitsunari	Cybozu Labs, Inc., Tokyo, Japan
Eiji Okamoto	LCIS, University of Tsukuba, Japan
Tadanori Teruya	LCIS, University of Tsukuba, Japan

Outline of the talk

1. Context
2. Hardware accelerator for the Tate pairing over supersingular curves
3. Software accelerator for the Tate pairing over supersingular curves
4. Optimal Ate Pairing over Barreto-Naehrig Curves

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$
- A non-degenerate bilinear pairing is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$$

that satisfies the following conditions:

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$
- A non-degenerate bilinear pairing is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$$

  that satisfies the following conditions:
  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_\tau}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_\tau$)

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$
- A non-degenerate bilinear pairing is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\tau$$

  that satisfies the following conditions:
  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_\tau}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_\tau$)
  - bilinearity:
    $\hat{e}(Q_1 + Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R) \quad \hat{e}(Q, R_1 + R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$
- A non-degenerate bilinear pairing is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_\tau$$

  that satisfies the following conditions:

  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_\tau}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_\tau$)
  - bilinearity:
    $\hat{e}(Q_1+Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R) \quad \hat{e}(Q, R_1+R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$
  - computability: $\hat{e}$ can be efficiently computed

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$
- A non-degenerate bilinear pairing is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$$

  that satisfies the following conditions:
  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_\tau}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_\tau$)
  - bilinearity: $\hat{e}(Q_1 + Q_2, R) = \hat{e}(Q_1, R) \cdot \hat{e}(Q_2, R) \quad \hat{e}(Q, R_1 + R_2) = \hat{e}(Q, R_1) \cdot \hat{e}(Q, R_2)$
  - computability: $\hat{e}$ can be efficiently computed
- Immediate property: for any two integers k_1 and k_2

$$\hat{e}(k_1 Q, k_2 R) = \hat{e}(Q, R)^{k_1 k_2}$$

Bilinear pairings

- Let $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ be two additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = \ell$
- $(\mathbb{G}_\tau, \times)$, a multiplicatively-written cyclic group of order $\#\mathbb{G}_\tau = \ell$
- A non-degenerate bilinear pairing is a map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\tau$$

  that satisfies the following conditions:

  - non-degeneracy: $\hat{e}(P, P) \neq 1_{\mathbb{G}_\tau}$ (equivalently $\hat{e}(P, P)$ generates $\mathbb{G}_\tau$)
  - bilinearity:
    $\hat{e}(Q_1+Q_2, R) = \hat{e}(Q_1, R)\cdot\hat{e}(Q_2, R) \quad \hat{e}(Q, R_1+R_2) = \hat{e}(Q, R_1)\cdot\hat{e}(Q, R_2)$
  - computability: $\hat{e}$ can be efficiently computed

- Immediate property: for any two integers k_1 and k_2

$$\hat{e}(k_1 Q, k_2 R) = \hat{e}(Q, R)^{k_1 k_2}$$

- When $\mathbb{G}_1 = \mathbb{G}_2$ we say that the pairing is symmetric, otherwise if $\mathbb{G}_1 \neq \mathbb{G}_2$, the pairing is asymmetric.

Pairings in cryptography

- At first, used to attack supersingular elliptic curves
  - Menezes-Okamoto-Vanstone and Frey-Rück attacks, 1993 and 1994

$$\begin{array}{ccc} \mathrm{DLP}_{\mathbb{G}_1} & <_{\mathrm{P}} & \mathrm{DLP}_{\mathbb{G}_\tau} \\ kP & \longrightarrow & \hat{e}(kP, P) = \hat{e}(P, P)^k \end{array}$$

  - for cryptographic applications, we will also require the DLP in $\mathbb{G}_\tau$ to be hard

Pairings in cryptography

- At first, used to attack supersingular elliptic curves
  - Menezes-Okamoto-Vanstone and Frey-Rück attacks, 1993 and 1994

$$\begin{array}{ccc} \mathrm{DLP}_{\mathbb{G}_1} & <_{\mathrm{P}} & \mathrm{DLP}_{\mathbb{G}_\tau} \\ kP & \longrightarrow & \hat{e}(kP, P) = \hat{e}(P, P)^k \end{array}$$

  - for cryptographic applications, we will also require the DLP in $\mathbb{G}_\tau$ to be hard
- One-round three-party key agreement (Joux, 2000)

Pairings in cryptography

- At first, used to attack supersingular elliptic curves
  - Menezes-Okamoto-Vanstone and Frey-Rück attacks, 1993 and 1994

$$\begin{array}{ccc} \mathrm{DLP}_{\mathbb{G}_1} & <_{\mathrm{P}} & \mathrm{DLP}_{\mathbb{G}_\tau} \\ kP & \longrightarrow & \hat{e}(kP, P) = \hat{e}(P, P)^k \end{array}$$

  - for cryptographic applications, we will also require the DLP in $\mathbb{G}_\tau$ to be hard
- One-round three-party key agreement (Joux, 2000)
- Identity-based encryption
  - Boneh–Franklin, 2001
  - Sakai–Kasahara, 2001

Pairings in cryptography

- At first, used to attack supersingular elliptic curves
  - Menezes-Okamoto-Vanstone and Frey-Rück attacks, 1993 and 1994

$$\begin{array}{ccc} \mathrm{DLP}_{\mathbb{G}_1} & <_{\mathrm{P}} & \mathrm{DLP}_{\mathbb{G}_\tau} \\ kP & \longrightarrow & \hat{e}(kP, P) = \hat{e}(P, P)^k \end{array}$$

  - for cryptographic applications, we will also require the DLP in $\mathbb{G}_\tau$ to be hard
- One-round three-party key agreement (Joux, 2000)
- Identity-based encryption
  - Boneh–Franklin, 2001
  - Sakai–Kasahara, 2001
- Short digital signatures
  - Boneh–Lynn–Shacham, 2001
  - Zang–Safavi-Naini–Susilo, 2004
- ...

The Tate Pairing over Supersingular elliptic curves

- We first define
  - $\mathbb{F}_q$, a finite field, with $q = 2^m$ or 3^m
  - E, an elliptic curve defined over $\mathbb{F}_q$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_q)$

The Tate Pairing over Supersingular elliptic curves

- We first define
  - $\mathbb{F}_q$, a finite field, with $q = 2^m$ or 3^m
  - E, an elliptic curve defined over $\mathbb{F}_q$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_q)$
- $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$, the $\mathbb{F}_q$-rational ℓ-torsion of E:

$$\mathbb{G}_1 = \{P \in E(\mathbb{F}_q) \mid \ell P = \mathcal{O}\}$$

The Tate Pairing over Supersingular elliptic curves

- We first define
  - $\mathbb{F}_q$, a finite field, with $q = 2^m$ or 3^m
  - E, an elliptic curve defined over $\mathbb{F}_q$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_q)$
- $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$, the $\mathbb{F}_q$-rational ℓ-torsion of E:

$$\mathbb{G}_1 = \{P \in E(\mathbb{F}_q) \mid \ell P = \mathcal{O}\}$$

- $\mathbb{G}_\tau = \mu_\ell$, the group of ℓ-th roots of unity in $\mathbb{F}_{q^k}^\times$:

$$\mathbb{G}_\tau = \{U \in \mathbb{F}_{q^k}^\times \mid U^\ell = 1\}$$

The Tate Pairing over Supersingular elliptic curves

- We first define
  - $\mathbb{F}_q$, a finite field, with $q = 2^m$ or 3^m
  - E, an elliptic curve defined over $\mathbb{F}_q$
  - ℓ, a large prime factor of $\#E(\mathbb{F}_q)$
- $\mathbb{G}_1 = E(\mathbb{F}_q)[\ell]$, the $\mathbb{F}_q$-rational ℓ-torsion of E:

$$\mathbb{G}_1 = \{P \in E(\mathbb{F}_q) \mid \ell P = \mathcal{O}\}$$

- $\mathbb{G}_\tau = \mu_\ell$, the group of ℓ-th roots of unity in $\mathbb{F}_{q^k}^\times$:

$$\mathbb{G}_\tau = \{U \in \mathbb{F}_{q^k}^\times \mid U^\ell = 1\}$$

- k is the embedding degree, the smallest integer such that $\mu_\ell \subseteq \mathbb{F}_{q^k}^\times$
  - usually large for ordinary elliptic curves
  - bounded in the case of supersingular elliptic curves
    (4 in characteristic 2; 6 in characteristic 3; and 2 in characteristic > 3)

Security considerations for Symmetric Pairings

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}^{\times}_{p^{km}}$$

- The discrete logarithm problem should be hard in both $\mathbb{G}_1$ and $\mathbb{G}_\tau$

Security considerations for Symmetric Pairings

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^{\times}$$

- The discrete logarithm problem should be hard in both $\mathbb{G}_1$ and $\mathbb{G}_T$

Base field ($\mathbb{F}_{p^m}$)	$\mathbb{F}_{2^m}$	$\mathbb{F}_{3^m}$
Lower security ($\sim 2^{64}$)	$m = 239$	$m = 97$
Medium security ($\sim 2^{80}$)	$m = 373$	$m = 163$
Higher security ($\sim 2^{128}$)	$m = 1103$	$m = 503$

- $\mathbb{F}_{2^m}$: simpler finite field arithmetic
- $\mathbb{F}_{3^m}$: smaller field extension

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}_{p^{km}}^{\times}$$

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}^{\times}_{p^{km}}$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \to \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$
- Arithmetic over $\mathbb{F}_{p^{km}}^\times$:
  - tower-field representation
  - only arithmetic over the underlying field $\mathbb{F}_{p^m}$

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \to \mu_\ell \subseteq \mathbb{F}_{p^{km}}^\times$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$
- Arithmetic over $\mathbb{F}_{p^{km}}^\times$:
  - tower-field representation
  - only arithmetic over the underlying field $\mathbb{F}_{p^m}$
- Operations over $\mathbb{F}_{p^m}$:
  - $O(m)$ additions / subtractions
  - $O(m)$ multiplications
  - $O(m)$ Frobenius maps ($a \mapsto a^p$, *i.e.* squarings or cubings)
  - 1 inversion

Computation of the Tate pairing

$$\hat{e} : E(\mathbb{F}_{p^m})[\ell] \times E(\mathbb{F}_{p^m})[\ell] \rightarrow \mu_\ell \subseteq \mathbb{F}^{\times}_{p^{km}}$$

- Arithmetic over $\mathbb{F}_{p^m}$:
  - polynomial basis: $\mathbb{F}_{p^m} \cong \mathbb{F}_p[x]/(f(x))$
  - $f(x)$, degree-m polynomial irreducible over $\mathbb{F}_p$
- Arithmetic over $\mathbb{F}^{\times}_{p^{km}}$:
  - tower-field representation
  - only arithmetic over the underlying field $\mathbb{F}_{p^m}$
- Operations over $\mathbb{F}_{p^m}$:
  - $O(m)$ additions / subtractions
  - $O(m)$ multiplications
  - $O(m)$ Frobenius maps ($a \mapsto a^p$, *i.e.* squarings or cubings)
  - 1 inversion
- A first idea: an all-in-one unified operator:
  - shared resources
  - scalable architecture

Motivations

- High speed is more important than low resources for some cryptographic applications
- Explore the other end of the area vs. time tradeoff:
  - faster but larger than the unified operator
  - what about the area-time product?

Motivations

- High speed is more important than low resources for some cryptographic applications
- Explore the other end of the area vs. time tradeoff:
  - faster but larger than the unified operator
  - what about the area-time product?
- Accelerate the computation by extracting as much parallelism as possible...
- ... Without increasing dramatically the resource requirements

Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

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

Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

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

Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q) = \eta_T(P, Q)^M$$

Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q) = \eta_T(P, Q)^M$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$

Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q) = \eta_T(P, Q)^M$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$
- Final exponentiation by $M = (p^{km} - 1)/N$
  - required to obtain a unique value for each congruence class
  - example in characteristic 3 ($k = 6$ and $N = 3^m + 1 \pm 3^{(m+1)/2}$):

$$M = \frac{3^{6m} - 1}{3^m + 1 \pm 3^{(m+1)/2}} = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

  - exploit the special form of the exponent: *ad-hoc* algorithm

Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps

$$\hat{e}(P, Q) = \eta_T(P, Q)^M$$

- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m + 1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}^{\times}_{p^{km}}$, with $N = \#E(\mathbb{F}_{p^m})$
- Final exponentiation by $M = (p^{km} - 1)/N$
  - required to obtain a unique value for each congruence class
  - example in characteristic 3 ($k = 6$ and $N = 3^m + 1 \pm 3^{(m+1)/2}$):

$$M = \frac{3^{6m} - 1}{3^m + 1 \pm 3^{(m+1)/2}} = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

  - exploit the special form of the exponent: *ad-hoc* algorithm
- Two distinct computational requirements

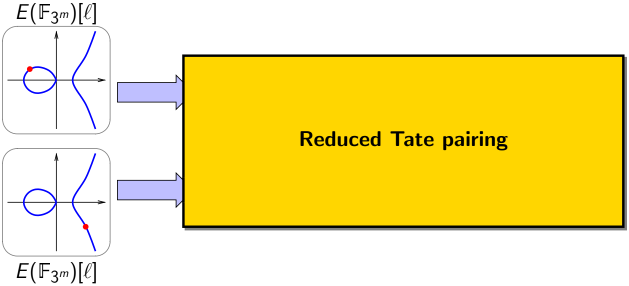
Computation of the η_T pairing

- The Tate pairing over $E(\mathbb{F}_{p^m})$ is computed in two main steps
$$\hat{e}(P,Q) = \eta_T(P,Q)^M$$
- Computation of the η_T pairing
  - via Miller's algorithm: loop of $(m+1)/2$ iterations
  - result only defined modulo N-th powers in $\mathbb{F}_{p^{km}}^{\times}$, with $N = \#E(\mathbb{F}_{p^m})$
- Final exponentiation by $M = (p^{km}-1)/N$
  - required to obtain a unique value for each congruence class
  - example in characteristic 3 ($k = 6$ and $N = 3^m + 1 \pm 3^{(m+1)/2}$):
$$M = \frac{3^{6m}-1}{3^m + 1 \pm 3^{(m+1)/2}} = \left(3^{3m}-1\right)\left(3^m+1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$
  - exploit the special form of the exponent: *ad-hoc* algorithm
- Two distinct computational requirements $\Rightarrow$ use two distinct coprocessors

Reduced Tate pairing

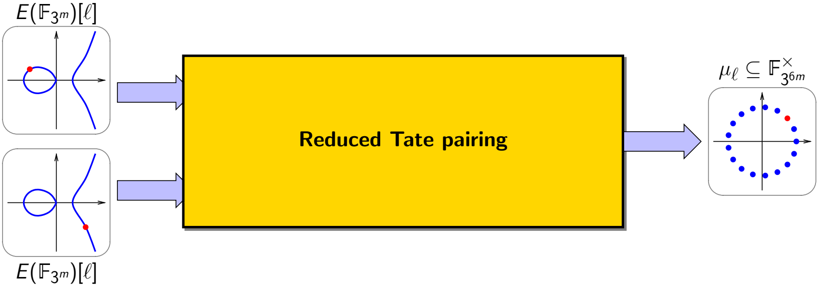
Reduced Tate pairing

Reduced Tate pairing



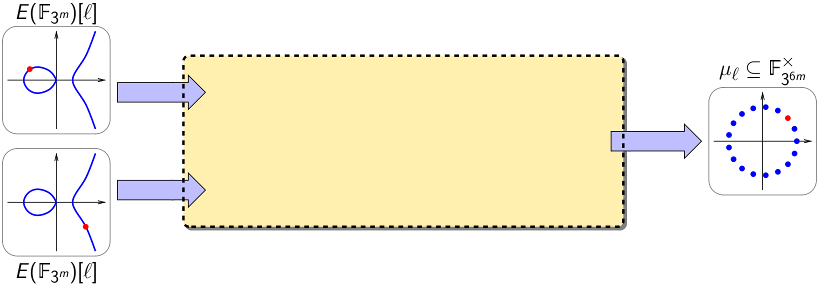
- Input: two points P and Q in $E(\mathbb{F}_{3^m})[\ell]$

Reduced Tate pairing



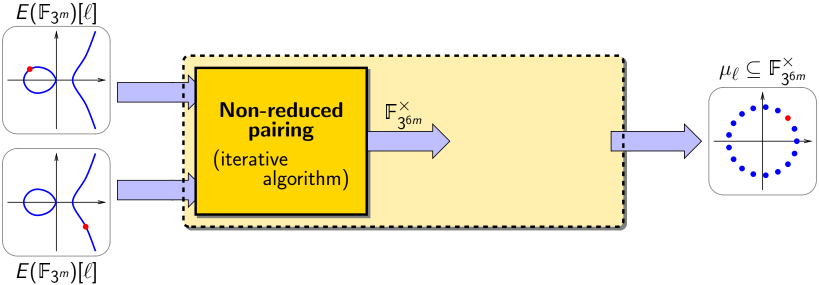
- Input: two points P and Q in $E(\mathbb{F}_{3^m})[\ell]$
- Output: an ℓ-th root of unity in the extension $\mathbb{F}^\times_{3^{6m}}$

Reduced Tate pairing



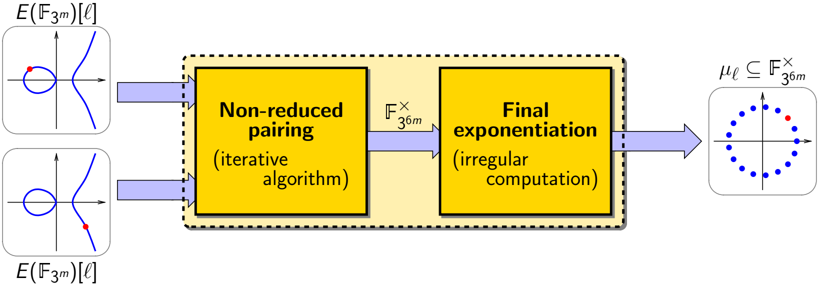
- Input: two points P and Q in $E(\mathbb{F}_{3^m})[\ell]$
- Output: an ℓ-th root of unity in the extension $\mathbb{F}^\times_{3^{6m}}$
- Two very different steps

Reduced Tate pairing



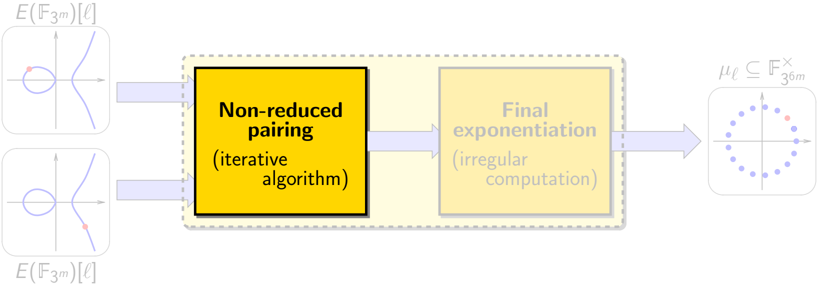
- Input: two points P and Q in $E(\mathbb{F}_{3^m})[\ell]$
- Output: an ℓ-th root of unity in the extension $\mathbb{F}_{3^{6m}}^{\times}$
- Two very different steps

Reduced Tate pairing



- Input: two points P and Q in $E(\mathbb{F}_{3^m})[\ell]$
- Output: an ℓ-th root of unity in the extension $\mathbb{F}^{\times}_{3^{6m}}$
- Two very different steps

Reduced Tate pairing



- Input: two points P and Q in $E(\mathbb{F}_{3^m})[\ell]$
- Output: an ℓ-th root of unity in the extension $\mathbb{F}_{3^{6m}}^\times$
- Two very different steps

Two coprocessors for the η_T pairing

- The two operations are purely sequential
- Only one active coprocessor at every moment

Two coprocessors for the η_T pairing

- The two operations are purely sequential
- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors

Two coprocessors for the η_T pairing

- The two operations are purely sequential
- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors
  - both of them are kept busy
  - higher throughput

Two coprocessors for the η_T pairing

- The two operations are purely sequential
- Only one active coprocessor at every moment
- Pipeline the data between the two coprocessors
  - both of them are kept busy
  - higher throughput
- Balance the computation time between the two coprocessors

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

- Three tasks per iteration:
  1. update the coordinates
  2. compute the line equation
  3. accumulate the new factor
- Total cost: $17\times$, 4 Frobenius/inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$

η_T pairing algorithm

$$\eta_T : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \rightarrow \mathbb{F}_{3^{6m}}^{\times}$$

- Three tasks per iteration:
  - ① update the coordinates
  - ② compute the line equation
  - ③ accumulate the new factor
- Total cost: $17\times$, 4 Frobenius/inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$
- Cost of the inverse Frobenius: Same as the Frobenius

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	2 inv. Frobenius 2 Frobenius	$(\mathbb{F}_{3^m})$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times$, $1+$	$(\mathbb{F}_{3^m})$
③	$R \leftarrow R \cdot S$	$15\times$, $29+$	$(\mathbb{F}_{3^m})$

end for

Accelerating the η_T pairing

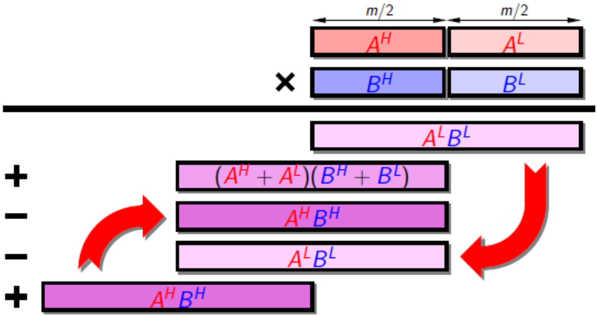
- Total cost: $17\times$, 2 Frobenius and inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius/inverse Frobenius and $+$: cheap and fast operations

Accelerating the η_T pairing

- Total cost: $17\times$, 2 Frobenius and inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius/inverse Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba

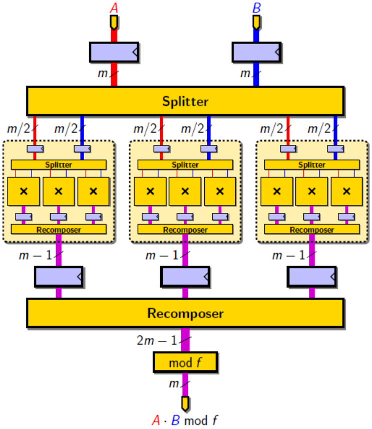
Accelerating the η_T pairing

- Total cost: $17\times$, 2 Frobenius and inverse Frobenius and $30+$ over $\mathbb{F}_{3^m}$ per iteration
  - Frobenius/inverse Frobenius and $+$: cheap and fast operations
  - critical operation: $\times$
- Need for a fast parallel multiplier: Karatsuba



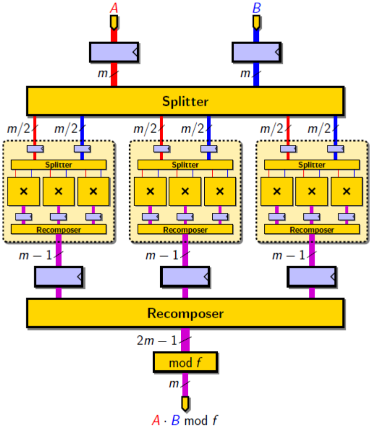
$$A^H B^L + A^L B^H = (A^H + A^L)(B^H + B^L) - A^H B^H - A^L B^L$$

A parallel Karatsuba multiplier



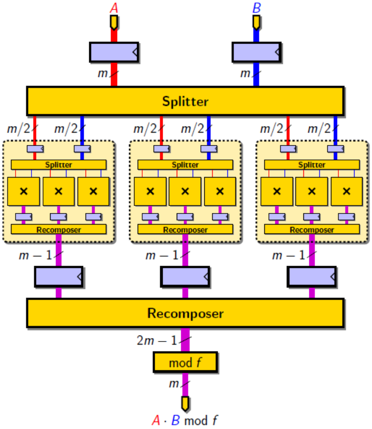
- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle

A parallel Karatsuba multiplier



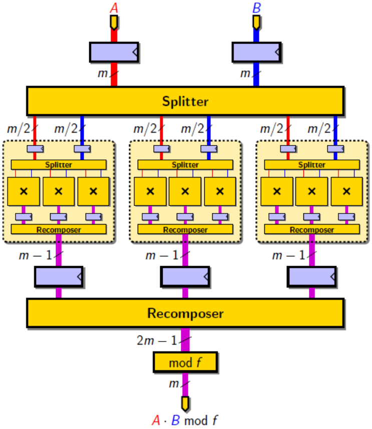
- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle
- sub-products recursively implemented as Karatsuba-Ofman multipliers

A parallel Karatsuba multiplier



- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle
- sub-products recursively implemented as Karatsuba-Ofman multipliers
- support for other variants: odd-even split, 3-way split, ...

A parallel Karatsuba multiplier



- fully parallel: all sub-products are computed in parallel
- pipelined architecture: higher clock frequency, one product per cycle
- sub-products recursively implemented as Karatsuba-Ofman multipliers
- support for other variants: odd-even split, 3-way split, ...
- final reduction modulo the irreducible polynomial f

Accelerating the $\boldsymbol{\eta_T}$ pairing

- η_T coprocessor based on a single large multiplier:
  - parallel Karatsuba architecture
  - 7-stage pipeline
  - one product per cycle

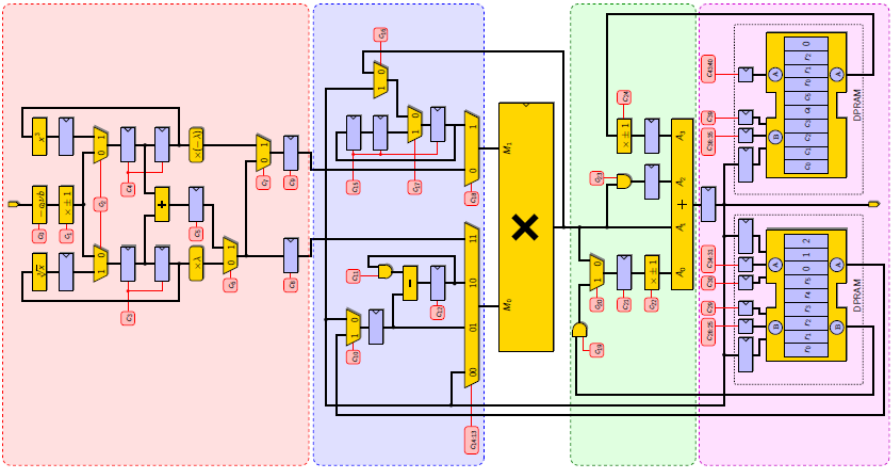
Accelerating the η_T pairing

- η_T coprocessor based on a single large multiplier:
  - parallel Karatsuba architecture
  - 7-stage pipeline
  - one product per cycle
- Challenge: keep the multiplier busy at all times

Accelerating the η_T pairing

- η_T coprocessor based on a single large multiplier:
  - parallel Karatsuba architecture
  - 7-stage pipeline
  - one product per cycle
- Challenge: keep the multiplier busy at all times
- Careful scheduling to avoid pipeline bubbles (idle cycles):
  - ensure that multiplication operands are always available
  - avoid memory congestion issues
- We managed to accomplish that: our processor computes Miller loop in just $17 \cdot (m+3)/2$ clock cycles (considering the initialization phase)

A parallel operator for the η_T pairing



The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^{\times}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^{\times}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\,\times$, $3m + 3$ Frobenius, $3m + 175\,+$, and 1 inversion

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}^{\times}_{3^{6m}}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\,\times$, $3m + 3$ Frobenius, $3m + 175\,+$, and 1 inversion ($\sim \log m \times$ and $m - 1$ Frobenius)

The final exponentiation

- Compute $\hat{e}(P,Q)$ as $\eta_T(P,Q)^M$ with $\eta_T(P,Q) \in \mathbb{F}^{\times}_{3^{6m}}$ and

$$M = \left(3^{3m}-1\right)\left(3^m+1\right)\left(3^m+1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73\,\times$, $3m+3$ Frobenius, $3m+175\,+$, and 1 inversion ($\sim \log m\,\times$ and $m-1$ Frobenius)
- Cost of the η_T pairing:
  - $(m+1)/2$ iterations
  - $17\,\times$, 10 Frobenius and $30\,+$ over $\mathbb{F}_{3^m}$ per iteration

The final exponentiation

- Compute $\hat{e}(P, Q)$ as $\eta_T(P, Q)^M$ with $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}^{\times}$ and

$$M = \left(3^{3m} - 1\right)\left(3^m + 1\right)\left(3^m + 1 \mp 3^{(m+1)/2}\right)$$

- Operations over $\mathbb{F}_{3^m}$: $73 \times$, $3m + 3$ Frobenius, $3m + 175 +$, and 1 inversion ($\sim \log m \times$ and $m - 1$ Frobenius)
- Cost of the η_T pairing:
  - $(m + 1)/2$ iterations
  - $17 \times$, 10 Frobenius and $30 +$ over $\mathbb{F}_{3^m}$ per iteration
- The final exponentiation is much cheaper than the η_T pairing
- Challenge for the final exponentiation:
  - computation in the same time as the η_T pairing
  - ... using as few resources as possible

The final exponentiation

- Design the smallest architecture possible supporting all the required operations over $\mathbb{F}_{3^m}$
- purely sequential scheduling

The final exponentiation

- Design the smallest architecture possible supporting all the required operations over $\mathbb{F}_{3^m}$
- purely sequential scheduling
- Although some parallelism is required.

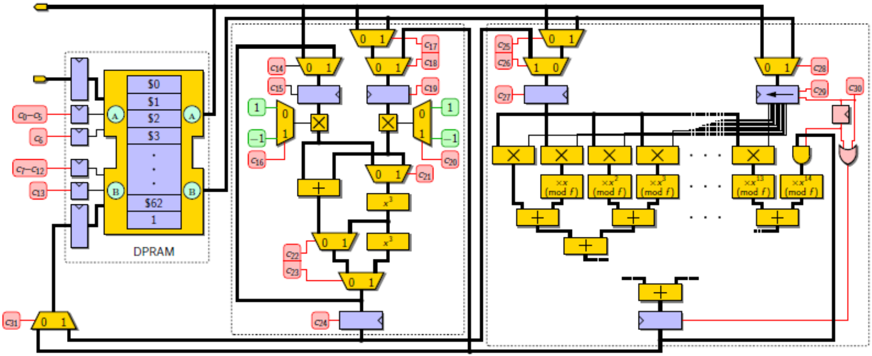
The final exponentiation

- Design the smallest architecture possible supporting all the required operations over $\mathbb{F}_{3^m}$
- purely sequential scheduling
- Although some parallelism is required.
- We found out that the usage of the inverse Frobenius operator is advantageous for computing the final exponentiation (as long as the irreducible polynomials are inverse-Frobenius friendly)

The final exponentiation

- Design the smallest architecture possible supporting all the required operations over $\mathbb{F}_{3^m}$
- purely sequential scheduling
- Although some parallelism is required.
- We found out that the usage of the inverse Frobenius operator is advantageous for computing the final exponentiation (as long as the irreducible polynomials are inverse-Frobenius friendly)
- New coprocessor with two arithmetic units:
  - a standalone multiplier, based on a parallel-serial scheme
  - a unified operator supporting addition/subtraction, inverse Frobenius map and inverse double Frobenius map

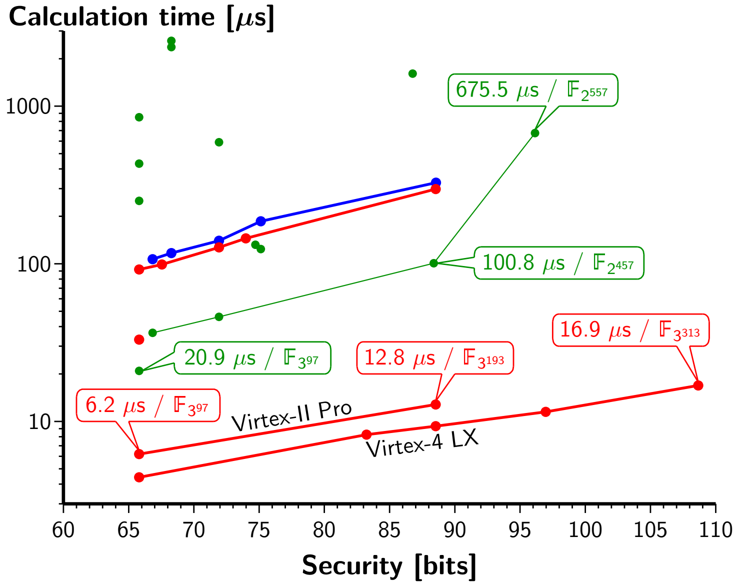
A coprocessor for the final exponentiation



Agenda

1. Context
2. Hardware accelerator for the Tate pairing over supersingular curves
   - Implementation Results in Hardware
3. Software accelerator for the Tate pairing over supersingular curves
   - Computing the non-reduced pairing
   - Final exponentiation
   - Implementation results
4. Optimal Ate Pairing over Barreto-Naehrig Curves
   - Barreto–Naehrig Curves

Hardware accelerators



Hardware implementation notes

- Our Xilinx FPGA implementation, significantly improved the computation time of all the hardware pairing coprocessors for supersingular curves previously published

Hardware implementation notes

- Our Xilinx FPGA implementation, significantly improved the computation time of all the hardware pairing coprocessors for supersingular curves previously published
- (a bit Surprisingly) our architecture also enjoys the best area/time trade-off performance among supersingular pairing accelerators

Hardware implementation notes

- Our Xilinx FPGA implementation, significantly improved the computation time of all the hardware pairing coprocessors for supersingular curves previously published
- (a bit Surprisingly) our architecture also enjoys the best area/time trade-off performance among supersingular pairing accelerators
- However, because we exceeded the FPGA's capacity, we could only achieve up to 109 bits of security

Hardware implementation notes

- Our Xilinx FPGA implementation, significantly improved the computation time of all the hardware pairing coprocessors for supersingular curves previously published
- (a bit Surprisingly) our architecture also enjoys the best area/time trade-off performance among supersingular pairing accelerators
- However, because we exceeded the FPGA's capacity, we could only achieve up to 109 bits of security
- Although it was not discussed here, we also implemented the Tate pairing over char 2. Experimentally, we observed that our char 2 and char 3 accelerators achieve almost the same time performance

Hardware implementation notes

- Our Xilinx FPGA implementation, significantly improved the computation time of all the hardware pairing coprocessors for supersingular curves previously published
- (a bit Surprisingly) our architecture also enjoys the best area/time trade-off performance among supersingular pairing accelerators
- However, because we exceeded the FPGA's capacity, we could only achieve up to 109 bits of security
- Although it was not discussed here, we also implemented the Tate pairing over char 2. Experimentally, we observed that our char 2 and char 3 accelerators achieve almost the same time performance
- In the design process of our char 2 accelerator we found the following undocumented family of square-root friendly irreducible pentanomials: $f(x) = x^m + x^{m-d} + x^{m-2d} + x^d + 1$.
- all technical details of these designs can be found in the preprint manuscripts eprint 2009/122 and eprint 2009/398

Agenda

1. Context
2. Hardware accelerator for the Tate pairing over supersingular curves
   - Implementation Results in Hardware
3. Software accelerator for the Tate pairing over supersingular curves
   - Computing the non-reduced pairing
   - Final exponentiation
   - Implementation results
4. Optimal Ate Pairing over Barreto-Naehrig Curves
   - Barreto–Naehrig Curves

Computing the non-reduced pairing

- η_T pairing: shorter loop

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

$$x_P \leftarrow \sqrt[3]{x_P} \quad ; \quad y_P \leftarrow \sqrt[3]{y_P}$$
$$x_Q \leftarrow x_Q^3 \quad ; \quad y_Q \leftarrow y_Q^3$$

$$t \leftarrow x_P + x_Q \qquad u \leftarrow y_P y_Q$$
$$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$$

$$R \leftarrow R \cdot S$$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ — $2\ \sqrt[3]{\cdot}$

$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$ — $2\ (\cdot)^3$

$t \leftarrow x_P + x_Q \qquad u \leftarrow y_P y_Q$

$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$

$R \leftarrow R \cdot S$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates
  - ② computation of line equation

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

① $x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ — $2\ \sqrt[3]{\cdot}$
$x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$ — $2\ (\cdot)^3$

② $t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ — $2\times, 2+$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$

$R \leftarrow R \cdot S$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates
  - ② computation of line equation
  - ③ accumulation of the new factor

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$1\times\ (\mathbb{F}_{3^{6m}})$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates
  - ② computation of line equation
  - ③ accumulation of the new factor
- Multiplication is critical
- Comb right-to-left multiplier over $\mathbb{F}_{3^m}$

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$1\times\ (\mathbb{F}_{3^{6m}})$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates
  - ② computation of line equation
  - ③ accumulation of the new factor
- Multiplication is critical
- Comb right-to-left multiplier over $\mathbb{F}_{3^m}$
- Sparse multiplication over $\mathbb{F}_{3^{6m}}$

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$1\times\ (\mathbb{F}_{3^{6m}})$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates
  - ② computation of line equation
  - ③ accumulation of the new factor
- Multiplication is critical
- Comb right-to-left multiplier over $\mathbb{F}_{3^m}$
- Sparse multiplication over $\mathbb{F}_{3^{6m}}$
  - 15 $\times$ and 29 $+$ over $\mathbb{F}_{3^m}$ (Beuchat *et al.*, ARITH 18)

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$15\times, 29+$

end for

Computing the non-reduced pairing

- η_T pairing: shorter loop
- Based on Miller's algorithm:
  - ① update of point coordinates
  - ② computation of line equation
  - ③ accumulation of the new factor
- Multiplication is critical
- Comb right-to-left multiplier over $\mathbb{F}_{3^m}$

for $i \leftarrow 0$ **to** $(m-1)/2$ **do**

①	$x_P \leftarrow \sqrt[3]{x_P}$; $y_P \leftarrow \sqrt[3]{y_P}$ $x_Q \leftarrow x_Q^3$; $y_Q \leftarrow y_Q^3$	$2\ \sqrt[3]{\cdot}$ $2\ (\cdot)^3$
②	$t \leftarrow x_P + x_Q$; $u \leftarrow y_P y_Q$ $S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$	$2\times, 2+$
③	$R \leftarrow R \cdot S$	$12\times, 59+$

end for

- Sparse multiplication over $\mathbb{F}_{3^{6m}}$
  - $15\times$ and $29+$ over $\mathbb{F}_{3^m}$ (Beuchat *et al.*, ARITH 18)
  - $12\times$ and $59+$ over $\mathbb{F}_{3^m}$ (Gorla *et al.*, SAC 2007)

Computing the non-reduced pairing

First core

for $i \leftarrow 1$ **to** $(m-1)/2$ **do**

① $x_P[i] \leftarrow \sqrt[3]{x_P[i-1]}$; $y_P[i] \leftarrow \sqrt[3]{y_P[i-1]}$ — $2\ \sqrt[3]{\cdot}$
$x_Q[i] \leftarrow x_Q[i-1]^3$; $y_Q[i] \leftarrow y_Q[i-1]^3$ — $2\ (\cdot)^3$

end for

for $i \leftarrow 1$ **to** $(m-1)/2$ **do**

② $t \leftarrow x_P[i] + x_Q[i]$ — $1\,+$
$u \leftarrow y_P[i] y_Q[i]$ — $1\,\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\,\times, 1\,+$

③ $R \leftarrow R \cdot S$ — $12\,\times, 59\,+$

end for

Computing the non-reduced pairing

First core

for $i \leftarrow 1$ **to** $(m-1)/2$ **do**

① $x_P[i] \leftarrow \sqrt[3]{x_P[i-1]}$; $y_P[i] \leftarrow \sqrt[3]{y_P[i-1]}$ $2\ \sqrt[3]{\cdot}$
$x_Q[i] \leftarrow x_Q[i-1]^3$; $y_Q[i] \leftarrow y_Q[i-1]^3$ $2\ (\cdot)^3$

end for

for $i \leftarrow 1$ **to** $(m-1)/4$ **do**

② $t \leftarrow x_P[i] + x_Q[i]$ $1+$
$u \leftarrow y_P[i]y_Q[i]$ $1\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ $1\times, 1+$

③ $R_0 \leftarrow R_0 \cdot S$ $12\times, 59+$

end for

Second core

for $i \leftarrow (m-1)/4+1$ **to** $(m-1)/2$ **do**

② $t \leftarrow x_P[i] + x_Q[i]$ $1+$
$u \leftarrow y_P[i]y_Q[i]$ $1\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ $1\times, 1+$

③ $R_1 \leftarrow R_1 \cdot S$ $12\times, 59+$

end for

Computing the non-reduced pairing

First core

for $i \leftarrow 1$ **to** $(m-1)/2$ **do**

① $x_P[i] \leftarrow \sqrt[3]{x_P[i-1]}$; $y_P[i] \leftarrow \sqrt[3]{y_P[i-1]}$ — $2\ \sqrt[3]{\cdot}$
$x_Q[i] \leftarrow x_Q[i-1]^3$; $y_Q[i] \leftarrow y_Q[i-1]^3$ — $2\ (\cdot)^3$

end for

for $i \leftarrow 1$ **to** $(m-1)/4$ **do**

② $t \leftarrow x_P[i] + x_Q[i]$ — $1+$
$u \leftarrow y_P[i] y_Q[i]$ — $1\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\times, 1+$

③ $R_0 \leftarrow R_0 \cdot S$ — $12\times, 59+$

end for

$R \leftarrow R_0 \cdot R_1$ — $15\times, 67+$

Second core

for $i \leftarrow (m-1)/4 + 1$ **to** $(m-1)/2$ **do**

② $t \leftarrow x_P[i] + x_Q[i]$ — $1+$
$u \leftarrow y_P[i] y_Q[i]$ — $1\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\times, 1+$

③ $R_1 \leftarrow R_1 \cdot S$ — $12\times, 59+$

end for

Computing the non-reduced pairing

First core

for $i \leftarrow 1$ **to** $(m-1)/2$ **do**

① $x_P[i] \leftarrow \sqrt[3]{x_P[i-1]}$; $y_P[i] \leftarrow \sqrt[3]{y_P[i-1]}$ — $2\ \sqrt[3]{\cdot}$
$x_Q[i] \leftarrow x_Q[i-1]^3$; $y_Q[i] \leftarrow y_Q[i-1]^3$ — $2\ (\cdot)^3$

end for

for $i \leftarrow 1$ **to** $(m-1)/8$ **do**

② $t \leftarrow x_P[2i-1] + x_Q[2i-1]$ — $1\,+$
$u \leftarrow y_P[2i-1]y_Q[2i-1]$ — $1\,\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\,\times, 1\,+$

③ $R_0 \leftarrow R_0 \cdot S$ — $12\,\times, 59\,+$

② $t \leftarrow x_P[2i] + x_Q[2i]$ — $1\,+$
$u \leftarrow y_P[2i]y_Q[2i]$ — $1\,\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\,\times, 1\,+$

③ $R_0 \leftarrow R_0 \cdot S$ — $12\,\times, 59\,+$

end for

$R \leftarrow R_0 \cdot R_1$ — $15\,\times, 67\,+$

Second core

for $i \leftarrow (m-1)/8+1$ **to** $(m-1)/4$ **do**

② $t \leftarrow x_P[2i-1] + x_Q[2i-1]$ — $1\,+$
$u \leftarrow y_P[2i-1]y_Q[2i-1]$ — $1\,\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\,\times, 1\,+$

③ $R_1 \leftarrow R_1 \cdot S$ — $12\,\times, 59\,+$

② $t \leftarrow x_P[2i] + x_Q[2i]$ — $1\,+$
$u \leftarrow y_P[2i]y_Q[2i]$ — $1\,\times$
$S \leftarrow -t^2 \pm u\sigma - t\rho - \rho^2$ — $1\,\times, 1\,+$

③ $R_1 \leftarrow R_1 \cdot S$ — $12\,\times, 59\,+$

end for

Computing the non-reduced pairing

First core

for $i \leftarrow 1$ **to** $(m-1)/2$ **do**

① $x_P[i] \leftarrow \sqrt[3]{x_P[i-1]}$; $y_P[i] \leftarrow \sqrt[3]{y_P[i-1]}$ — $2\ \sqrt[3]{\cdot}$
$x_Q[i] \leftarrow x_Q[i-1]^3$; $y_Q[i] \leftarrow y_Q[i-1]^3$ — $2\ (\cdot)^3$

end for

for $i \leftarrow 1$ **to** $(m-1)/8$ **do**

②
$t_0 \leftarrow x_P[2i-1] + x_Q[2i-1]$ — $1\,+$
$u_0 \leftarrow y_P[2i-1] y_Q[2i-1]$ — $1\,\times$
$t_1 \leftarrow x_P[2i] + x_Q[2i]$ — $1\,+$
$u_1 \leftarrow y_P[2i] y_Q[2i]$ — $1\,\times$
$S \leftarrow (-t_0^2 \pm u_0\sigma - t_0\rho - \rho^2)\cdot (-t_1^2 \pm u_1\sigma - t_1\rho - \rho^2)$ — $8\,\times, 13\,+$

③ $R_0 \leftarrow R_0 \cdot S$ — $15\,\times, 67\,+$

end for

$R \leftarrow R_0 \cdot R_1$ — $15\,\times, 67\,+$

Second core

for $i \leftarrow (m-1)/8 + 1$ **to** $(m-1)/4$ **do**

②
$t_0 \leftarrow x_P[2i-1] + x_Q[2i-1]$ — $1\,+$
$u_0 \leftarrow y_P[2i-1] y_Q[2i-1]$ — $1\,\times$
$t_1 \leftarrow x_P[2i] + x_Q[2i]$ — $1\,+$
$u_1 \leftarrow y_P[2i] y_Q[2i]$ — $1\,\times$
$S \leftarrow (-t_0^2 \pm u_0\sigma - t_0\rho - \rho^2)\cdot (-t_1^2 \pm u_1\sigma - t_1\rho - \rho^2)$ — $8\,\times, 13\,+$

③ $R_1 \leftarrow R_1 \cdot S$ — $15\,\times, 67\,+$

end for

Agenda

Final exponentiation

- Final exponentiation consists of raising $\hat{e}(P, Q)$ to the exponent,

$$M = \frac{2^{4m} - 1}{N} = (2^{2m} - 1) \cdot (2^m + 1 - \nu 2^{(m+1)/2}),$$

  where $\nu = (-1)^b$ when $m \equiv 1, 7 \pmod 8$ and $\nu = (-1)^{1-b}$ in all other cases.
- Highly sequential computation, Very heterogeneous

Final exponentiation

- Final exponentiation consists of raising $\hat{e}(P, Q)$ to the exponent,

$$M = \frac{2^{4m}-1}{N} = (2^{2m}-1)\cdot(2^m+1-\nu 2^{(m+1)/2}),$$

where $\nu = (-1)^b$ when $m \equiv 1, 7 \pmod 8$ and $\nu = (-1)^{1-b}$ in all other cases.

- Highly sequential computation, Very heterogeneous
- We perform this operation according to a slightly optimized version:
  - **Raising to the $(2^m + 1)$-th power.** Raising the outcome of Miller's algorithm to the $\left(2^{2m}-1\right)$-th power produces an element $U \in \mathbb{F}_{2^{4m}}$ of order $2^{2m}+1$. This property allows one to save a multiplication over $\mathbb{F}_{2^{4m}}$ when raising U to the (2^m+1)-th power.

Final exponentiation

- Final exponentiation consists of raising $\hat{e}(P, Q)$ to the exponent,

$$M = \frac{2^{4m} - 1}{N} = (2^{2m} - 1) \cdot (2^m + 1 - \nu 2^{(m+1)/2}),$$

  where $\nu = (-1)^b$ when $m \equiv 1, 7 \pmod 8$ and $\nu = (-1)^{1-b}$ in all other cases.
- Highly sequential computation, Very heterogeneous
- We perform this operation according to a slightly optimized version:
  - **Raising to the $(\mathbf{2^m+1})$-th power.** Raising the outcome of Miller's algorithm to the $\left(2^{2m} - 1\right)$-th power produces an element $U \in \mathbb{F}_{2^{4m}}$ of order $2^{2m} + 1$. This property allows one to save a multiplication over $\mathbb{F}_{2^{4m}}$ when raising U to the $(2^m + 1)$-th power.
  - **Raising to the $\mathbf{2^{\frac{m+1}{2}}}$-th power.** raising an element of $\mathbb{F}_{2^{4m}}$ to the 2^i-th power involves $4i$ squarings and at most four additions over $\mathbb{F}_{2^m}$

Final exponentiation

- Final exponentiation consists of raising $\hat{e}(P, Q)$ to the exponent,

$$M = \frac{2^{4m} - 1}{N} = (2^{2m} - 1) \cdot (2^m + 1 - \nu 2^{(m+1)/2}),$$

  where $\nu = (-1)^b$ when $m \equiv 1, 7 \pmod 8$ and $\nu = (-1)^{1-b}$ in all other cases.
- Highly sequential computation, Very heterogeneous
- We perform this operation according to a slightly optimized version:
  - **Raising to the $(2^m + 1)$-th power.** Raising the outcome of Miller's algorithm to the $\left(2^{2m} - 1\right)$-th power produces an element $U \in \mathbb{F}_{2^{4m}}$ of order $2^{2m} + 1$. This property allows one to save a multiplication over $\mathbb{F}_{2^{4m}}$ when raising U to the $(2^m + 1)$-th power.
  - **Raising to the $2^{\frac{m+1}{2}}$-th power.** raising an element of $\mathbb{F}_{2^{4m}}$ to the 2^i-th power involves $4i$ squarings and at most four additions over $\mathbb{F}_{2^m}$

Finite field arithmetic

- Target: multi-core architectures

Finite field arithmetic

- Target: multi-core architectures
- Arithmetic over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$: SSE instruction set

Finite field arithmetic

- Target: multi-core architectures
- Arithmetic over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$: SSE instruction set
- Timings are given in clock cycles and were measured on an Intel Core 2 processor working at 2.4 GHz.

	Field	x^p	$\sqrt[p]{x}$	Mult
Aranha et al. CT-RSA'10	$\mathbb{F}_{2^{1223}}$	160	166	4030
Our work CANS'10	$\mathbb{F}_{2^{1223}}$	480	749	5438
	$\mathbb{F}_{3^{509}}$	900	974	4128

Agenda

Implementation results

- Timings achieved on an Intel Core2 are given in millions of clock cycles
- Windows XP 64-bit SP2 environment

Implementation results

- Timings achieved on an Intel Core2 are given in millions of clock cycles
- Windows XP 64-bit SP2 environment

	Curve	Security [bits]	# of cores	Freq. [GHz]	Calc. time [Mcycles]
Aranha et al. CT-RSA'10	$E(\mathbb{F}_{2^{1223}})$	128	1	2.4	18.76
	$E(\mathbb{F}_{2^{1223}})$	128	2	2.4	10.08
	$E(\mathbb{F}_{2^{1223}})$	128	4	2.4	5.72
Our work CANS'10	$E(\mathbb{F}_{3^{509}})$	128	1	2.4	18.2
	$E(\mathbb{F}_{3^{509}})$	128	2	2.4	10.34
	$E(\mathbb{F}_{3^{509}})$	128	4	2.4	7.06

Software implementation notes: The supersingular case

- Significantly faster implementation (for a while)

Software implementation notes: The supersingular case

- Significantly faster implementation (for a while)
- How many cores?

Software implementation notes: The supersingular case

- Significantly faster implementation (for a while)
- How many cores?
  - acceleration always less than the ideal speedup factor
  - best choice: dual-core implementation

Software implementation notes: The supersingular case

- Significantly faster implementation (for a while)
- How many cores?
  - acceleration always less than the ideal speedup factor
  - best choice: dual-core implementation
- Characteristic 3 performs better than characteristic 2

Software implementation notes: The supersingular case

- Significantly faster implementation (for a while)
- How many cores?
  - acceleration always less than the ideal speedup factor
  - best choice: dual-core implementation
- Characteristic 3 performs better than characteristic 2
  - at least on Intel Core2 and Intel Core i7

Software implementation notes: The supersingular case

- Significantly faster implementation (for a while)
- How many cores?
  - acceleration always less than the ideal speedup factor
  - best choice: dual-core implementation
- Characteristic 3 performs better than characteristic 2
  - at least on Intel Core2 and Intel Core i7
  - next generation of processors: built-in carry-less 64-bit multiplier
  - the battle is not over!

Agenda

1. Context
2. Hardware accelerator for the Tate pairing over supersingular curves
   - Implementation Results in Hardware
3. Software accelerator for the Tate pairing over supersingular curves
   - Computing the non-reduced pairing
   - Final exponentiation
   - Implementation results
4. Optimal Ate Pairing over Barreto-Naehrig Curves
   - Barreto–Naehrig Curves

Barreto–Naehrig Curves

Defined by the equation $E : y^2 = x^3 + b$, where $b \neq 0$. Their embedding degree k is equal to 12. The characteristic p of the prime field, the group order r, and the trace of Frobenius t_r of the curve are parametrized as follows:

$$\begin{aligned} p(t) &= 36t^4 + 36t^3 + 24t^2 + 6t + 1, \\ r(t) &= 36t^4 + 36t^3 + 18t^2 + 6t + 1, \\ t_r(t) &= 6t^2 + 1, \end{aligned} \qquad (1)$$

where $t \in \mathbb{Z}$ is an arbitrary integer such that $p = p(t)$ and $r = r(t)$ are both prime numbers.

For efficiency purposes, t must have a low Hamming weight. In this work we used,

$$t = 2^{62} - 2^{54} + 2^{44}$$

Barreto–Naehrig Curves

Let $E[r]$ denote the r-torsion subgroup of E and π_p be the Frobenius endomorphism $\pi_p : E \to E$ given by $\pi_p(x, y) = (x^p, y^p)$. We define,

- $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$,
- $\mathbb{G}_2 \subseteq E(\mathbb{F}_{p^{12}})[r]$,
- $\mathbb{G}_T = \mu_r \subset \mathbb{F}^*_{p^{12}}$ (*i.e.* the group of r-th roots of unity).
- The optimal ate pairing on the BN curve E is given as,

$$\begin{aligned} a_{\text{opt}} : \mathbb{G}_2 \times \mathbb{G}_1 &\longrightarrow \mathbb{G}_3 \\ (Q, P) &\longmapsto \Big(f_{6t+2,Q}(P) \cdot l_{[6t+2]Q,\pi_p(Q)}(P) \cdot \\ &\qquad l_{[6t+2]Q+\pi_p(Q),-\pi_p^2(Q)}(P)\Big)^{\frac{p^{12}-1}{r}}, \end{aligned}$$

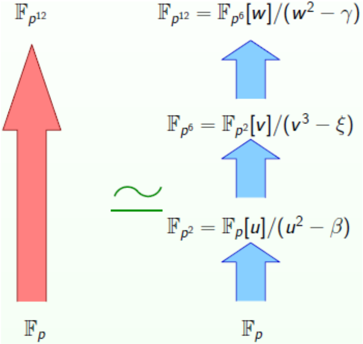
- In practice, pairing computations can be restricted to points P and Q' that belong to $E(\mathbb{F}_p)$ and $E'(\mathbb{F}_{p^2})$, respectively, where, $E'/\mathbb{F}_{p^2} : y^2 = x^3 + b/\xi$.

Optimal ate pairing algorithm

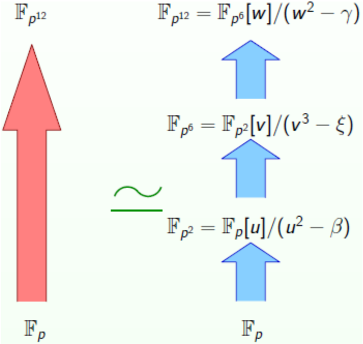
Input: $P \in \mathbb{G}_1$ y $Q \in \mathbb{G}_2$.
Output: $a_{\text{opt}}(Q, P)$.

Write $s = 6t + 2$ as $s = \sum_{i=0}^{L-1} s_i 2^i$, where $s_i \in \{-1, 0, 1\}$;
$T \leftarrow Q,\ f \leftarrow 1$;
for $i = L - 2$ to 0 **do**
  $f \leftarrow f^2 \cdot l_{T,T}(P);\ T \leftarrow 2T$;
  if $s_i = -1$ **then**
    $f \leftarrow f \cdot l_{T,-Q}(P);\ T \leftarrow T - Q$;
  else if $s_i = 1$ **then**
    $f \leftarrow f \cdot l_{T,Q}(P);\ T \leftarrow T + Q$;
  end if
end for
$Q_1 \leftarrow \pi_p(Q);\ \ Q_2 \leftarrow \pi_{p^2}(Q)$;
$f \leftarrow f \cdot l_{T,Q_1}(P);\ T \leftarrow T + Q_1$;
$f \leftarrow f \cdot l_{T,-Q_2}(P);\ T \leftarrow T - Q_2$;
$f \leftarrow f^{(p^{12}-1)/r}$;
return f;

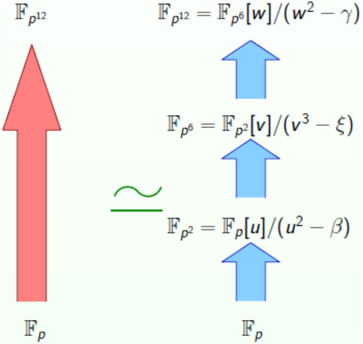
Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.



Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.

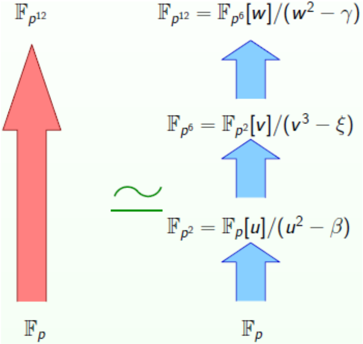


Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.



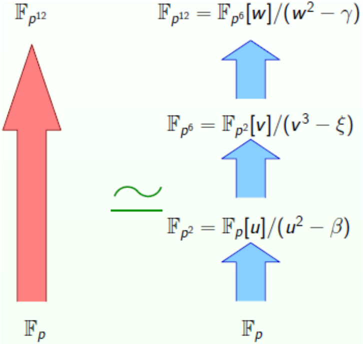
$$\beta = -5$$

Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.



$$\begin{aligned} \xi &= u \\ \beta &= -5 \end{aligned}$$

Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.

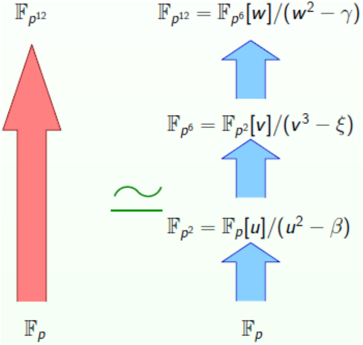


$$\gamma = v$$

$$\xi = u$$

$$\beta = -5$$

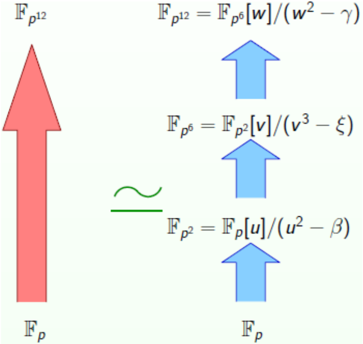
Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.



$f = g + hw \in \mathbb{F}_{p^{12}}$,
with $g, h \in \mathbb{F}_{p^6}$.

but also
$g = g_0 + g_1 v + g_2 v^2$,
$h = h_0 + h_1 v + h_2 v^2$,
where $g_i, h_i \in \mathbb{F}_{p^2}$,
for $i = 1, 2, 3$.

Since $p \bmod 12 \equiv 1$ we can build the towering up to the twelfth extension by adjoining irreducible binomial only.



hence, we can write $f \in \mathbb{F}_{p^{12}}$ as

$$
\begin{aligned}
f &= g + hw \\
&= g + hw \\
&= g_0 + h_0 W + g_1 W^2 + \\
&\quad h_1 W^3 + g_2 W^4 + h_2 W^5.
\end{aligned}
$$

- Let (a, m, s, i), $(\tilde{a}, \tilde{m}, \tilde{s}, \tilde{i})$, and (A, M, S, I) denote the cost of field addition, multiplication, squaring, and inversion in $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, and $\mathbb{F}_{p^6}$, respectively
- we sometimes need to compute the multiplication in the base field by the constant coefficient $\beta \in \mathbb{F}_p$ of the irreducible binomial $f(u) = u^2 - \beta$. We refer to this operation as m_β
- we sometimes need to compute the multiplication of an arbitrary element in $\mathbb{F}_{p^2}$ times the constant $\xi = u \in \mathbb{F}_p$ at a cost of one multiplication by the constant β. We refer to this operation as m_ξ, but it is noticed that the cost of m_ξ is essentially the same of that of m_β.

Computational costs of the tower extension field arithmetic

Field	Add/Sub	Mult	Squaring	Inversion
$\mathbb{F}_{p^2}$	$\tilde{a} = 2a$	$\tilde{m} = 3m + 3a + m_\beta$	$\tilde{s} = 2m + 3a + m_\beta$	$\tilde{i} = 4m + m_\beta$ $+2a + i$
$\mathbb{F}_{p^6}$	$3\tilde{a}$	$6\tilde{m} + 2m_\xi + 15\tilde{a}$	$2\tilde{m} + 3\tilde{s} + 2m_\xi + 8\tilde{a}$	$9\tilde{m} + 3\tilde{s} + 4m_\xi$ $+4\tilde{a} + \tilde{i}$
$\mathbb{F}_{p^{12}}$	$6\tilde{a}$	$18\tilde{m} + 6m_\xi + 60\tilde{a}$	$12\tilde{m} + 4m_\xi + 45\tilde{a}$	$25\tilde{m} + 9\tilde{s} + 12m_\xi$ $+61\tilde{a} + \tilde{i}$
$\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$	$6\tilde{a}$	$18\tilde{m} + 6m_\xi + 60\tilde{a}$	$9\tilde{s} + 4m_\xi$ $+30\tilde{a}$	Conjugate

We took advantage of the following design decisions,

- The bit-length of $6t + 2$ is $L = 65$

We took advantage of the following design decisions,

- The bit-length of $6t + 2$ is $L = 65$
  - This implies that we require 64 point doubling in the Miller loop.

We took advantage of the following design decisions,

- The bit-length of $6t + 2$ is $L = 65$
  - This implies that we require 64 point doubling in the Miller loop.
- The Hamming weight of $6t + 2$ is 7

We took advantage of the following design decisions,

- The bit-length of $6t + 2$ is $L = 65$
  - This implies that we require 64 point doubling in the Miller loop.
- The Hamming weight of $6t + 2$ is 7
  - This implies that we require 6 point addition/subtraction in the Miller loop.

We took advantage of the following design decisions,

- The bit-length of $6t + 2$ is $L = 65$
  - This implies that we require 64 point doubling in the Miller loop.
- The Hamming weight of $6t + 2$ is 7
  - This implies that we require 6 point addition/subtraction in the Miller loop.
- The low Hamming weight of t allows us to save arithmetic operations in the hard part of the final exponentiation

Mille Loop Cost

$$
\begin{aligned}
\text{Miller Loop} &= 64 \cdot (28\tilde{m} + 8\tilde{s} + 100\tilde{a} + 4m + 6m_\beta) + \\
&\quad 6 \cdot (20\tilde{m} + 7\tilde{s} + 64\tilde{a} + 4m + 2m_\beta) + \\
&\quad 40\tilde{m} + 14\tilde{s} + 128\tilde{a} + 14m + 4m_\beta \\
&= 1952\tilde{m} + 568\tilde{s} + 6912\tilde{a} + 294m + 400m_\beta.
\end{aligned}
$$

Calculating the final Exponentiation

We must compute $f \in \mathbb{F}_{p^{12}}$ raised to the power $e = (p^{12} - 1)/r$

Calculating the final Exponentiation

We must compute $f \in \mathbb{F}_{p^{12}}$ raised to the power $e = (p^{12} - 1)/r$

$$e = \frac{p^{12} - 1}{r} = (p^6 - 1) \cdot (p^2 + 1) \cdot \frac{p^4 - p^2 + 1}{r}.$$

Calculating the final Exponentiation

We must compute $f \in \mathbb{F}_{p^{12}}$ raised to the power $e = (p^{12} - 1)/r$

$$e = \frac{p^{12} - 1}{r} = (p^6 - 1) \cdot (p^2 + 1) \cdot \frac{p^4 - p^2 + 1}{r}.$$

- Raising to $f^{(p^6-1)} = \bar{f} \cdot f^{-1}$ costs one conjugation, one inversion and one multiplication over $\mathbb{F}_{p^{12}}$.
- After this step, f becomes an element of the cyclotomic group $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$.

Calculating the final Exponentiation

We must compute $f \in \mathbb{F}_{p^{12}}$ raised to the power $e = (p^{12} - 1)/r$

$$e = \frac{p^{12} - 1}{r} = (p^6 - 1) \cdot (p^2 + 1) \cdot \frac{p^4 - p^2 + 1}{r}.$$

- Raising to $f^{(p^6-1)} = \bar{f} \cdot f^{-1}$ costs one conjugation, one inversion and one multiplication over $\mathbb{F}_{p^{12}}$.
- After this step, f becomes an element of the cyclotomic group $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$.
- Raising to the power $p^2 + 1$ costs 5 multiplications over $\mathbb{F}_p$, and one multiplication over $\mathbb{F}_{p^{12}}$.

Calculating the final Exponentiation

We must compute $f \in \mathbb{F}_{p^{12}}$ raised to the power $e = (p^{12} - 1)/r$

$$e = \frac{p^{12} - 1}{r} = (p^6 - 1) \cdot (p^2 + 1) \cdot \frac{p^4 - p^2 + 1}{r}.$$

- Raising to $f^{(p^6-1)} = \bar{f} \cdot f^{-1}$ costs one conjugation, one inversion and one multiplication over $\mathbb{F}_{p^{12}}$.
- After this step, f becomes an element of the cyclotomic group $\mathbb{G}_{\Phi_6}(\mathbb{F}_{p^2})$.
- Raising to the power $p^2 + 1$ costs 5 multiplications over $\mathbb{F}_p$, and one multiplication over $\mathbb{F}_{p^{12}}$.
- Raising to the power $m^{(p^4-p^2+1)/r}$ is referred as the hard part of the final exponentiation

Hard part of the final exponentiation

We used the addition chain proposed by Scott et al. at Pairing'09

$$m^t,\ m^{t^2},\ m^{t^3},\ m^p,\ m^{p^2},\ m^{p^3},\ m^{(tp)},\ m^{(t^2p)},\ m^{(t^3p)},\ m^{(t^2p^2)},$$

Hard part of the final exponentiation

We used the addition chain proposed by Scott et al. at Pairing'09

$$m^t,\ m^{t^2},\ m^{t^3},\ m^p,\ m^{p^2},\ m^{p^3},\ m^{(tp)},\ m^{(t^2p)},\ m^{(t^3p)},\ m^{(t^2p^2)},$$

- Taking advantage of the Frobenius, we can easily compute, m^p, m^{p^2}, m^{p^3}, $m^{(tp)}$, $m^{(t^2p)}$, $m^{(t^3p)}$, y $m^{(t^2p^2)}$ at a cost of 35 multiplications in the base field $\mathbb{F}_p$.

Hard part of the final exponentiation

We used the addition chain proposed by Scott et al. at Pairing'09

$$m^t,\ m^{t^2},\ m^{t^3},\ m^p,\ m^{p^2},\ m^{p^3},\ m^{(tp)},\ m^{(t^2p)},\ m^{(t^3p)},\ m^{(t^2p^2)},$$

- Taking advantage of the Frobenius, we can easily compute, m^p, m^{p^2}, m^{p^3}, $m^{(tp)}$, $m^{(t^2p)}$, $m^{(t^3p)}$, y $m^{(t^2p^2)}$ at a cost of 35 multiplications in the base field $\mathbb{F}_p$.
- The most costly part of this procedure consists on the computation of $m^t, m^{t^2} = (m^t)^t, m^{t^3} = (m^{t^2})^t$.
- Since $t = 2^{62} - 2^{54} + 2^{44}$, these exponentiations can be computed at a cost of $62 \cdot 3 = 186$ cyclotomic squarings plus $2 \cdot 3 = 6$ multiplications over $\mathbb{F}_{p^{12}}$.

Final exponentiation computational cost

$$\begin{aligned}
\text{Exp. Final} &= (25\tilde{m} + 9\tilde{s} + 12m_\beta + 61\tilde{a} + \tilde{i}) + (18\tilde{m} + 6m_\beta + 60\tilde{a}) + \\
&\quad (18\tilde{m} + 6m_\beta + 60\tilde{a}) + 10m + \\
&\quad 13 \cdot (18\tilde{m} + 6m_\beta + 60\tilde{a}) + 4 \cdot (9\tilde{s} + 4m_\beta + 30\tilde{a}) + 70m + \\
&\quad 186 \cdot (9\tilde{s} + 4m_\beta + 30\tilde{a}) + 6 \cdot (18\tilde{m} + 6m_\beta + 60\tilde{a}) \\
&= 403\tilde{m} + 1719\tilde{s} + 7021\tilde{a} + 80m + 898m_\beta + \tilde{i}.
\end{aligned}$$

A Comparison of arithmetic operations required by the computation of the ate pairing variants.

		$\tilde{m}$	$\tilde{s}$	$\tilde{a}$	$\tilde{i}$	m_ξ
Hankerson et al. [Book chapter, 2007] R-ate pairing	Miller Loop	2277	356	6712	1	412
	Final Exp.	1616	1197	8977	1	1062
	Total	3893	1553	15689	2	1474
Naehrig et al. [LatinCrypt 2010] Optimal ate pairing	Miller Loop	2022	590	7140		410
	Final Exp.	678	1719	7921	1	988
	Total	2700	2309	15061	1	1398
This work [Pairing 2010] Optimal ate pairing	Miller Loop	1954	568	6912		400
	Final Exp	443	1719	7021	1	898
	Total	2397	2287	13933	1	1298

Library Implementation

- We use the mul operation included in the x86-64 instruction set. It multiplies two 64-bit unsigned integers in about 3 clock cycles on Intel Core i7 and AMD Opteron processors
- An element $x \in \mathbb{F}_p$ is represented as $x = (x_3, x_2, x_1, x_0)$, where $x_i, 0 \leq i \leq 3$, are 64-bit integers
- Multiplication and inversion over $\mathbb{F}_p$ are accomplished according to the well-known Montgomery multiplication and Montgomery inversion algorithms, respectively
- The 256-bit integer multiplication and Montgomery reduction are computed in 55 and 100 clock cycles, respectively

Cycle counts of multiplication over $\mathbb{F}_{p^2}$, squaring over $\mathbb{F}_{p^2}$, and optimal ate pairing on different machines

	Our results			
	Core i7[a]	Opteron[b]	Core 2 Duo[c]	Athlon 64 X2[d]
Multiplication over $\mathbb{F}_{p^2}$	435	443	558	473
Squaring over $\mathbb{F}_{p^2}$	342	355	445	376
Miller loop	1,330,000	1,360,000	1,680,000	1,480,000
Final exponentiation	1,000,000	1,040,000	1,270,000	1,150,000
Optimal ate pairing	2,330,000	2,400,000	2,950,000	2,630,000

[a] Intel Core i7 860 (2.8GHz), Windows 7, Visual Studio 2008 Professional

[b] Quad-Core AMD Opteron 2376 (2.3GHz), Linux 2.6.18, gcc 4.4.1

[c] Intel Core 2 Duo T7100 (1.8GHz), Windows 7, Visual Studio 2008 Professional

[d] Athlon 64 X2 Dual Core 6000+(3GHz), Linux 2.6.23, gcc 4.1.2

[e] Intel Core 2 Quad Q6600 (2394MHz), Linux 2.6.28, gcc 4.3.3

Comparison Table

A comparison of cycles and timings required by the computation of the ate pairing variants. The frequency is given in GHz and the timings are in milliseconds.

	Alg.	Architecture	Cycles	Freq.	Calc. time
Aranha *et al.* [CT-RSA 2010]	η_T	Intel Xeon 45nm (1 core)	17,400,000	2.0	8.70
		Intel Xeon 45nm (8 cores)	3,020,000		1.51
Beuchat *et al.* [CANS 2009]	η_T	Intel Core i7 (1 core)	15,138,000	2.9	5.22
		Intel Core i7 (8 cores)	5,423,000		1.87
Hankerson *et al.*	R-ate	Intel Core 2	10,000,000	2.4	4.10
Naehrig *et al.* eprint 2010/526, April.6.2010	a_{opt}	Intel Core 2 Quad Q6600	4,470,000	2.4	1.80
Fan *et al.* CHES'09	"R-ate"	130 nm ASIC	59,976	.204	2.91
This Work eprint 2010/526, jun.17.2010	a_{opt}	Intel Core i7	2,330,000	2.8	0.83
Aranha et al. eprint 2010/526, oct.19.2010	a_{opt}	Intel Core i7	1,703,000	2.8	0.608

Software Implementation notes on ordinary curves

- records do not last long in software!

Software Implementation notes on ordinary curves

- records do not last long in software!
- However, in hardware the performance records hold longer: Fan *et al.* CHES'09 and Beuchat *et al.* CHES'09 are still the fastest hardware accelerators for ordinary and supersingular curves, respectively.

Software Implementation notes on ordinary curves

- records do not last long in software!
- However, in hardware the performance records hold longer: Fan *et al.* CHES'09 and Beuchat *et al.* CHES'09 are still the fastest hardware accelerators for ordinary and supersingular curves, respectively.
- Future projects/open problems,
  - To target higher security levels in software implementation of pairings (e.g, 192 bits of security)

Software Implementation notes on ordinary curves

- records do not last long in software!
- However, in hardware the performance records hold longer: Fan *et al.* CHES'09 and Beuchat *et al.* CHES'09 are still the fastest hardware accelerators for ordinary and supersingular curves, respectively.
- Future projects/open problems,
  - To target higher security levels in software implementation of pairings (e.g, 192 bits of security)
  - To design a hardware accelerator faster than any software library for asymmetric pairings over BN curves at 128-bit of security

Software Implementation notes on ordinary curves

- records do not last long in software!
- However, in hardware the performance records hold longer: Fan *et al.* CHES'09 and Beuchat *et al.* CHES'09 are still the fastest hardware accelerators for ordinary and supersingular curves, respectively.
- Future projects/open problems,
  - To target higher security levels in software implementation of pairings (e.g, 192 bits of security)
  - To design a hardware accelerator faster than any software library for asymmetric pairings over BN curves at 128-bit of security
  - to implement efficient pairing-based protocols in software and/or hardware

Software Implementation notes on ordinary curves

- records do not last long in software!
- However, in hardware the performance records hold longer: Fan *et al.* CHES'09 and Beuchat *et al.* CHES'09 are still the fastest hardware accelerators for ordinary and supersingular curves, respectively.
- Future projects/open problems,
  - To target higher security levels in software implementation of pairings (e.g, 192 bits of security)
  - To design a hardware accelerator faster than any software library for asymmetric pairings over BN curves at 128-bit of security
  - to implement efficient pairing-based protocols in software and/or hardware

Thank you for your attention

Questions?