Reducing the Key Size of Rainbow using Non-Commutative Rings

Takanori Yasuda (Institute of systems, Information Technologies and Nanotechnologies (ISIT)),
Kouichi Sakurai (ISIT, Kyushu university)
and Tsuyoshi Takagi (Kyushu university).

MQ problem and key size

- **MQ equations**

$$\begin{cases} f_1(x_1,\ldots,x_n) = \sum_{1\le i,j\le n} a_{ij}^{(1)} x_i x_j + \sum_{1\le i\le n} b_i^{(1)} x_i + c^{(1)} = d_1 \\ f_2(x_1,\ldots,x_n) = \sum_{1\le i,j\le n} a_{ij}^{(2)} x_i x_j + \sum_{1\le i\le n} b_i^{(2)} x_i + c^{(2)} = d_2 \\ \vdots \\ f_m(x_1,\ldots,x_n) = \sum_{1\le i,j\le n} a_{ij}^{(m)} x_i x_j + \sum_{1\le i\le n} b_i^{(m)} x_i + c^{(m)} = d_m \end{cases}$$

- **Public key = set of coefficients of polynomials**

$$\textit{the number of coefficients} = \frac{m(n+1)(n+2)}{2} \Rightarrow$$ large key size

Rainbow

- Signature of a multilayer variant of UOV
- Secret key size is also large

We reduced by 75%

?

Rainbow	Secret key size	Public key size	Key size of RSA	Ratio (secret key)
R(17,13,13)	19.1kB	25.7kB	1369bits	116.6times
R(21,16,17)	36.5kB	50.8kB	1937bits	150.7times
R(27,19,19)	60.5kB	84.0kB	2560bits	189.0times

Reference: A.Petzoldt et al. "Selecting Parameters for the Rainbow Signature Scheme", PQCrypto'10, Springer LNCS vol. 6061 (2010)

Reduction by 62%

- Problem: reduction of key size
  - reduction of public key
    CyclicRainbow (INDOCRIPT' 10, SCC' 10, PKC' 11)
  - reduction of secret key TTS, TRMS

Proposed scheme
Rainbow using non-commutative rings

$$\mathrm{NC\text{-}Rainbow}(R; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_s)$$

Non-commutative rings

- **R: non-commutative ring**
  - R: finite dimensional algebra over a finite field K (dimension=r)

  Fix a K-linear isomorphism $\phi : K^r \xrightarrow{\sim} R$

- **Example (quaternion algebra** Q_q (q: order of K))

$$\begin{cases} set) \quad Q_q = K \cdot 1 \oplus K \cdot i \oplus K \cdot j \oplus K \cdot ij, & (r = 4), \\ product) \quad i^2 = j^2 = -1, \quad ij = -ji. \end{cases}$$

There is a natural isomorphism $\phi : K^4 \xrightarrow{\sim} Q_q$

NC-Rainbow$(R;\tilde{v}_1,\tilde{o}_1,\tilde{o}_2,\ldots,\tilde{o}_s)$ (1/2)

$\tilde{n}$: positive number

$$0<\tilde{v}_1<\tilde{v}_2<\cdots<\tilde{v}_s<\tilde{v}_{s+1}=\tilde{n}$$

For $i = 1, \ldots, s$

- $\tilde{S}_i=\{1,\ldots,\tilde{v}_i\},\ \tilde{O}_i=\{\tilde{v}_i+1,\ldots,\tilde{v}_{i+1}\},$
- $\tilde{o}_i=\tilde{v}_{i+1}-\tilde{v}_i.$

$\boxed{\text{Central map}}$ $\tilde{G}=(\tilde{g}_{\tilde{v}_1+1},\ldots,\tilde{g}_n):R^{\tilde{n}}\to R^{\tilde{m}},\quad (\tilde{m}=\tilde{n}-\tilde{v}_1)$

$$\begin{aligned}\tilde{g}_k(x_1,\ldots,x_n)=&\sum_{i\in\tilde{O}_h,\,j\in\tilde{S}_h}(x_i\alpha_{ij}^{(k)}x_j+x_j\alpha_{ij}^{(k)}x_i)\\&+\sum_{i,j\in\tilde{S}_h}x_i\beta_{ij}^{(k)}x_j+\sum_{i\in\tilde{S}_{h+1}}(\gamma_i^{(k,1)}x_i+x_i\gamma_i^{(k,2)})\\&+\eta^{(k)}\qquad(k=\tilde{v}_1+1,\ldots,\tilde{n},\ \ \alpha_{ij}^{(k)},\ldots\in R).\end{aligned}$$

NC-Rainbow$(R;\tilde{v}_1,\tilde{o}_1,\tilde{o}_2,\ldots,\tilde{o}_s)$ (2/2)

- **Key Generation**
  - Secret key $(n=\tilde{n}r, m=\tilde{m}r)$
    - $\tilde{G}$, two affine transformations $A_1: K^m \to K^m$, $A_2: K^n \to K^n$.
  - Public key
    - $\tilde{F} = A_1 \circ \phi^{-\tilde{m}} \circ \tilde{G} \circ \phi^{\tilde{n}} \circ A_2 : K^n \to K^m$.
- **Signature Generation**

  For message $M \in K^m$, calculate

  $$(1)\ a=\phi^{\tilde{m}}(A_1^{-1}(M)),\ (2)\ b=\tilde{G}^{-1}(a),\ (3)\ c=\phi^{-\tilde{n}}(A_2^{-1}(b))$$

  in this order. *c* is a signature.
- **Verification**
  - If $\tilde{F}(c)=M$, the signature is accepted.

Original Rainbow

If *R=K* , then this becomes $\boxed{\mathrm{Rainbow}(K;\tilde{v}_1,\tilde{o}_1,\tilde{o}_2,\ldots,\tilde{o}_s)}$

Correspondence between NC-Rainbow and Rainbow

Theorem

There exists a correspondence

$$\text{NC-Rainbow}(R; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_s)$$
$$\to \text{Rainbow}(K; r\tilde{v}_1, r\tilde{o}_1, r\tilde{o}_2, \ldots, r\tilde{o}_s)$$

which holds public key.

- **Secret key size of NC-Rainbow**

$$m(m+1) + n(n+1) + \sum_{h=1}^{s} r\tilde{o}_h (2\tilde{v}_h \tilde{o}_h + \tilde{v}_h^2 + 2\tilde{v}_{h+1} + 1) \text{ field elements}$$

- **Secret key size of corresponding Rainbow**

$$m(m+1) + n(n+1) + \sum_{h=1}^{s} r\tilde{o}_h \left(r^2 \tilde{v}_h \tilde{o}_h + \frac{r\tilde{v}_h (r\tilde{v}_h + 1)}{2} + r\tilde{v}_{h+1} + 1 \right) \text{ field elements}$$

Comparison of Secret key size

$K = GF(256)$,

$R = Q_{256}$ $(r = 4)$.

Comparison of NC-Rainbow$(Q_{256}; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2)$ *and* Rainbow$(GF(256); 4\tilde{v}_1, 4\tilde{o}_1, 4\tilde{o}_2)$

$(\tilde{v}_1, \tilde{o}_1, \tilde{o}_2)$	NC-size	Corr. Rainbow	R-size	ratio
(4,3,3)	4.2kB	(16,12,12)	15.9kB	26.7%
(5,4,4)	8.0kB	(20,16,16)	33.6kB	23.9%
(7,5,5)	15.1kB	(28,20,20)	70.7kB	21.5%
(9,6,6)	25.5kB	(36,24,24)	128.2kB	19.9%

NC-size : Secret key size of NC-Rainbow

R-size : Secret key size of corresponding Rainbow

ratio = NC-size/R-size

Reason of reduction of key size

- **Property of "regular action"**
  - R is expressed by a subring of matrix algebra of size r.

$$Q_q \ni (\square \quad \square \quad \square \quad \square) \longrightarrow \begin{pmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{pmatrix} \in M(4, K)$$

4 entries 16 entries

$$M(d, Q_q) \longrightarrow M(4d, K)$$

$4d^2$ entries $16d^2$ entries

$$\mathrm{NC\text{-}Rainbow}(Q_q; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2, \ldots, \tilde{o}_s)$$

(Map in Theorem) $\rightarrow \mathrm{Rainbow}(K; 4\tilde{v}_1, 4\tilde{o}_1, 4\tilde{o}_2, \ldots, 4\tilde{o}_s)$

Attacks against Rainbow (1/2)

Need to analyze attacks against Rainbow to know whether or not they work efficiently against NC-Rainbow.

- **Known attacks against Rainbow**
  - Direct attacks
    - Using XL and Grobner basis algorithm etc.
  - UOV attack
    - determine a simultaneous isotropic subspace (which coincides with Oil space in the last layer with high probability)
    - compute invariant spaces of certain matrices
  - UOV-Reconciliation(UOV-R) attack
    - determine a simultaneous isotropic subspace (which coincides with Oil space in the last layer with high probability)
    - Solve a system of equations w.r.t. coefficients of right affine transformation

Attacks against Rainbow (2/2)

- **Known attacks against Rainbow (continued)**
  - MinRank attack
    - determine a matrix with minimal rank among linear combinations of quadratic part of components of public key
  - HighRank attack
    - determine a matrix with the second highest rank among linear combinations of quadratic part of components of public key
  - Rainbow-Band-Separation(RBS) attack
    - transform public key to a form of central map of Rainbow
    - Solve a system of equations w.r.t. coefficients of both affine transformations

Security for NC−Rainbow

l −bit security　(K=GF(2^a))

1. UOV attack

$$n - 2r\tilde{o}_s \geq l/a + 1$$

2. MinRank attack

$$r(\tilde{o}_1 + \tilde{v}_1) \geq l/a$$

3. HighRank attack

$$r\tilde{o}_s \geq l/a$$

4. UOV−R attack

$\tilde{v}_1 \geq \tilde{o}_1 \Rightarrow$ same security level against direct attacks

5. Direct attacks, RBS attack

A.Petzoldt et al. "Selecting Parameters for the Rainbow Signature Scheme", PQCrypto'10, Springer LNCS vol. 6061 (2010)

Table of security and secret key size

$$\text{NC-Rainbow}(Q_{256}; \tilde{v}_1, \tilde{o}_1, \tilde{o}_2)\ \textit{and}\ \text{Rainbow}(GF(256); 4\tilde{v}_1, 4\tilde{o}_1, 4\tilde{o}_2)$$

NC-Rainbow	$(5,4,4)$	$(7,5,5)$	$(9,6,6)$
Security level	83bits	96bits	107bits
Secret key size	8.0kB	15.1kB	25.5kB
Corr. Rainbow	(20,16,16)	(28,20,20)	(36,24,24)
Secret key size	33.6kB	70.7kB	128.2kB
Ratio	23.9%	21.5%	19.9%

ratio＝Secret key size of NCRainbow/Secret key size of corr. Rainbow

Conclusion

- **Conclusion**
  - We proposed a scheme using non-commutative rings, which is regarded as another construction of Rainbow.
  - This scheme can reduce the secret key size in comparison with original Rainbow.
  - In paticular, the secret key size of the proposed NC-Rainbow is reduced by about 75% in the security level of 80 bits.

- **Future works**
  - Finding a non-commutative ring with efficient arithmetic operation.
    - ⇒ Speed up the signature generation

Dual Exponentiation Schemes

Colin D. Walter

Information Security Group
Royal Holloway University of London

`Colin.Walter@rhul.ac.uk`

29 Feb 2012

RSACONFERENCE2012

The Problem

- *Motivation*: New algorithms are always useful as there are always so many different optimisations and conflicting pressures on resource-constrained platforms.
- *Aim*: Better exponentiation on space-limited chip. (Fast memory is expensive.)
- *Setting*: Mixed base representation for the exponent.
- *Solution*: Define a *dual* for the associated addition chain.
- *Benefits*: Derive new algorithms from existing ones; Better understanding of exponentiation.

Information Security Group

Outline

Information Security Group

Background

r-ary Exponentiation — L2R (Brauer, 1939)

Inputs: $g \in G$,
$D = ((d_{n-1}r+d_{n-2})r+\ldots+d_1)r+d_0 \in \mathbb{N}$ where $0 \le d_i < r$.

Output: $g^D \in G$

Initialise table: $T[d] \leftarrow g^d$ **for all** d, $0<d<r$.
$P \leftarrow 1_G$
for $i \leftarrow n-1$ **downto** 0 **do** {
  if $i \neq n-1$ **then** $P \leftarrow P^r$
  if $d_i \neq 0$ **then** $P \leftarrow P \times T[d_i]$ }
return P

Information Security Group

r-ary Exponentiation — R2L (Yao, 1976)

Inputs: $g \in G$,
$D = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \ldots + d_1 r^1 + d_0$ where $0 \le d_i < r$.
Output: $g^D \in G$

Initialise table: $T[d] \leftarrow 1_G$ **for all** d, $0<d<r$.
$P \leftarrow g$
for $i \leftarrow 0$ **to** $n-1$ **do** $\{$
  if $d_i \neq 0$ **then** $T[d_i] \leftarrow T[d_i] \times P$
  if $i \neq n-1$ **then** $P \leftarrow P^r$ $\}$
return $\prod_{d:0<d<r} T[d]^d$

Information Security Group

Sliding Window — L2R

Inputs: $g \in G$,
$D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$, where
$d_i \in \{0, \pm1, \pm3, \ldots, \pm\frac{1}{2}(r-1)\}$, $r_i \in \{2, 2^w\}$ and $d_i=0$ if $r_i=2$.

Output: $g^D \in G$

Initialise table: $T[d] \leftarrow g^d$ **for all** $d \neq 0$.
$P \leftarrow 1_G$
for $i \leftarrow n-1$ **downto** 0 **do** {
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$
  if $d_i \neq 0$ **then** $P \leftarrow P \times T[d_i]$ }
return P

Sliding Window — R2L

Inputs: $g \in G$,
$D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$, where
$d_i \in \{0, \pm1, \pm3, \ldots, \pm\frac{1}{2}(r-1)\}$, $r_i \in \{2, 2^w\}$ and $d_i=0$ if $r_i=2$.

Output: $g^D \in G$

Initialise table: $T[d] \leftarrow 1_G$ **for all** $d \neq 0$.
$P \leftarrow g$
for $i \leftarrow 0$ **to** $n-1$ **do** {
  if $d_i \neq 0$ **then** $T[d_i] \leftarrow T[d_i] \times P$
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$ }
return $\prod_{d\neq0} T[d]^d$

Information Security Group

Mixed Base Exponentiation — L2R

Inputs: $g \in G$,
$D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$,
where $(r_i, d_i) \in \mathcal{R}\times\mathcal{D}$.

Output: $g^D \in G$

Initialise table: $T[d] \leftarrow g^d$ **for all** $d \in \mathcal{D} \setminus \{0\}$.
$P \leftarrow 1_G$
for $i \leftarrow n-1$ **downto** 0 **do** $\{$
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$
  if $d_i \neq 0$ **then** $P \leftarrow P \times T[d_i]$ $\}$
return P

Information Security Group

Mixed Base Exponentiation — R2L

Inputs: $g \in G$,
$D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$,
where $(r_i, d_i) \in \mathcal{R}\times\mathcal{D}$.

Output: $g^D \in G$

Initialise table: $T[d] \leftarrow 1_G$ **for all** $d \in \mathcal{D} \setminus \{0\}$.
$P \leftarrow g$
for $i \leftarrow 0$ **to** $n-1$ **do** {
  if $d_i \neq 0$ **then** $T[d_i] \leftarrow T[d_i]\times P$
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$ }
return $\prod_{d\in\mathcal{D}\setminus\{0\}} T[d]^d$

A Compact Right-to-Left Algorithm (Arith13, 1997)

Inputs: $g \in G$,
$D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$,
where $(r_i, d_i) \in \mathcal{R}\times\mathcal{D}$.

Output: $g^D \in G$

$T \leftarrow 1_G$
$P \leftarrow g$
for $i \leftarrow 0$ **to** $n-1$ **do** $\{$
  if $d_i \neq 0$ **then** $T \leftarrow T \times P^{d_i}$
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$ $\}$
return T

The loop body involves computing P^{d_i} en route to P^{r_i}.

The Transposition Method

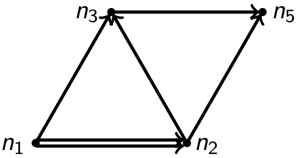
Information Security Group

The Computational Di-Graph

An addition chain for D yields a computational, acyclic *di-graph*:

Here is that for

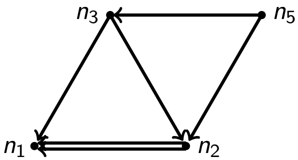
1+1=2; 1+2=3; 2+3=5.



For convenience, nodes are numbered so n_d represents g^d.

- Addition $i+j = k$ gives directed edges $n_i n_k$ and $n_j n_k$.
- It is *acyclic*, with a single root n_1 and a single leaf n_5.
- All nodes except root n_1 have input degree 2 as all op^s are binary.
- $\#\text{Ops} = \#\text{Nodes}-1 = \frac{1}{2}\#\text{Edges}$.
- By induction, $D = \#\text{paths from } n_1 \text{ to } n_D$.

Di-Graph for the Transpose Method



- Reverse the edges for the "*transposition*" method. Node inputs are again multiplied together.
- Path count is D, as before. So it again computes g^D.
- Nodes may need *merging* or *expanding* to restore in-degree 2. The #binary operations is not changed: $\frac{1}{2}$#edges.
- This reverses the addition chain in some sense.
- It *doesn't* preserve space requirements and without care, sq^{g} & mult^{n} counts may change.

Space Duality

Information Security Group

Space-Aware Addition Chains

Definition. For a given set of registers, take five classes of "atomic" op^s:

- *Copying* one register to another;
- *Copying* one register to another *& initialising* source register to 1_G;
- In-place *squaring* of the contents of one register;
- *Multiplying* two different registers into one of the input registers;
- *Multiplying* two different registers into one of the input registers, *& initialising* the other input to 1_G.

A **space-aware addition chain** is a sequence of such operations in which the registers are named.

Every addition chain can be written as a space-aware addition chain.

Matrix Representation — Space

For a device with two locations, matrix examples of each class are:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \text{ and } \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

They act on a column vector containing the values in each register.

By omitting more general op^{ns}, this set is *closed under transposition*.

- Copy (without initialise) becomes multiplication with initialise, and *vice versa*. (The *red* matrices.)
- Other operations stay in their class under transposition.

Definition. The *dual* of a space-aware chain is its transpose.
(The transposed operations are applied in reverse order.)

The dual uses the same space but may not have the same mult^{n} count.

Matrix Representation — Space

For a device with two locations, matrix examples of each class are:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

They act on a column vector containing the values in each register.

By omitting more general op^{ns}, this set is *closed under transposition*.

- Copy (without initialise) becomes multiplication with initialise, and *vice versa*. (The *red* matrices.)
- Other operations stay in their class under transposition.

Definition. The *dual* of a space-aware chain is its transpose.
(The transposed operations are applied in reverse order.)

The dual uses the same space but may not have the same mult^n count.

Matrix Representation — Space

For a device with two locations, matrix examples of each class are:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}.$$

They act on a column vector containing the values in each register.

By omitting more general op^{ns}, this set is *closed under transposition*.

- Copy (without initialise) becomes multiplication with initialise, and *vice versa*. (The *red* matrices.)
- Other operations stay in their class under transposition.

Definition. The *dual* of a space-aware chain is its transpose. (The transposed operations are applied in reverse order.)

The dual uses the same space but may not have the same mult^{n} count.

The Dual Chain — An Example

$R3 \leftarrow R2;\ R3 \leftarrow R2{+}R3;\ R1 \leftarrow_I R2;\ R2 \leftarrow_I R3;\ R2 \leftarrow_I R1{+}R2$

In matrices acting on a col^{mn} vector:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The dual (the transpose) is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

i.e. $R1 \leftarrow R2;\ R3 \leftarrow_I R2;\ R2 \leftarrow_I R1;\ R2 \leftarrow R2{+}R3;\ R2 \leftarrow_I R2{+}R3$

- Both have two multiplications and no squarings.
- Both compute g^3 from $g \in G$ with R_2 for I/O.

Extra Requirements

Information Security Group

The Main Problems

1. #Mults may not be preserved in the dual as copying becomes mult^n with initialisation.
2. The dual chain may not compute the same value unless the matrix product is symmetric.

To overcome the first of these, extra conditions are required:

- Select the initialising op^n when possible.
- Eliminate 1_G as an operand.
- Remove operations whose output is not used.
- Fix a subset of registers for I/O.
  (An I/O register *must* read input *and* write non-trivial output.)

Definition. A space-aware chain is *normalised* if the above hold.

The Main Problems

1. #Mults may not be preserved in the dual as copying becomes mult^{n} with initialisation.
2. The dual chain may not compute the same value unless the matrix product is symmetric.

To overcome the first of these, extra conditions are required:

- Select the initialising op^{n} when possible.
- Eliminate 1_G as an operand.
- Remove operations whose output is not used.
- Fix a subset of registers for I/O.
  (An I/O register *must* read input *and* write non-trivial output.)

Definition. A space-aware chain is *normalised* if the above hold.

The Main Problems

1. #Mults may not be preserved in the dual as copying becomes mult^n with initialisation.
2. The dual chain may not compute the same value unless the matrix product is symmetric.

To overcome the first of these, extra conditions are required:

- Select the initialising op^n when possible.
- Eliminate 1_G as an operand.
- Remove operations whose output is not used.
- Fix a subset of registers for I/O.
  (An I/O register *must* read input *and* write non-trivial output.)

Definition. A space-aware chain is *normalised* if the above hold.

Counting Ones

Instances of 1_G or $\perp$ arise from:

a) Initial value of a non-input register.

b) Initialised by copy or mult^n op^n.

Instances of 1_G or $\perp$ finish their lives as:

c) Final value in a non-output register.

d) Overwritten by a copy op^n.

Since $\#a = \#c$, we conclude $\#b = \#d$.
Subtracting the $\#\{\text{copies with init}^n\}$ from $\#b$ and $\#d$, we have

$$\#\text{Mult}^{ns} \text{ with init}^n = \#\text{Copies without init}^n$$

These op^n types are swapped in the dual & others stay as they are. So:

- **Theorem.** For a normalised space-aware chain,
  $\#\text{Mult}^{ns}$ & $\#\text{Sq}^{res}$ are the same for the dual.

Counting Ones

Instances of 1_G or $\perp$ arise from:

a) Initial value of a non-input register.

b) Initialised by copy or mult^n op^n.

Instances of 1_G or $\perp$ finish their lives as:

c) Final value in a non-output register.

d) Overwritten by a copy op^n.

Since $\#a = \#c$, we conclude $\#b = \#d$.
Subtracting the $\#\{\text{copies with init}^n\}$ from $\#b$ and $\#d$, we have

$$\#\text{Mult}^{ns} \text{ with init}^n = \#\text{Copies without init}^n$$

These op^n types are swapped in the dual & others stay as they are. So:

- **Theorem.** For a normalised space-aware chain, $\#\text{Mult}^{ns}$ & $\#\text{Sq}^{res}$ are the same for the dual.

Counting Ones

Instances of 1_G or $\perp$ arise from:

a) Initial value of a non-input register.

b) Initialised by copy or mult^n op^n.

Instances of 1_G or $\perp$ finish their lives as:

c) Final value in a non-output register.

d) Overwritten by a copy op^n.

Since $\#a = \#c$, we conclude $\#b = \#d$.
Subtracting the $\#\{\text{copies with init}^n\}$ from $\#b$ and $\#d$, we have

$$\#\text{Mult}^{ns} \text{ with init}^n = \#\text{Copies without init}^n$$

These op^n types are swapped in the dual & others stay as they are. So:

- **Theorem.** For a normalised space-aware chain, $\#\text{Mult}^{ns}$ & $\#\text{Sq}^{res}$ are the same for the dual.

Counting Ones

Instances of 1_G or $\perp$ arise from:

a) Initial value of a non-input register.

b) Initialised by copy or mult^n op^n.

Instances of 1_G or $\perp$ finish their lives as:

c) Final value in a non-output register.

d) Overwritten by a copy op^n.

Since $\#a = \#c$, we conclude $\#b = \#d$.
Subtracting the $\#\{$copies with init$^n\}$ from $\#b$ and $\#d$, we have

$$\#\text{Mult}^{\text{ns}} \text{ with init}^{\text{n}} = \#\text{Copies without init}^{\text{n}}$$

These op^n types are swapped in the dual & others stay as they are. So:

- **Theorem.** For a normalised space-aware chain, $\#\text{Mult}^{\text{ns}}$ & $\#\text{Sq}^{\text{res}}$ are the same for the dual.

Symmetric Cases

If the action of a (multi-) exponentiation function f on registers is described by matrix M then a dual f^* is described by the transpose M^{T}.

Theorem a) f^* computes the same values as f iff its matrix is symmetric.
b) In particular, it uses the same registers for output as input.

- In the normalised case, unused registers give columns of zeros.
- Used, non-output registers are over-written with 1_G: more zeros.
- Used, non-input registers are initialised to 1_G: more zeros.
- So only the sub-matrix M_{IO} on I/O registers need be symmetric.

Theorem A normalised space-aware chain for a *single* exponentiation and its dual compute the same values.

(Duals become unique only when written in atomic operations.)

Symmetric Cases

If the action of a (multi-) exponentiation function f on registers is described by matrix M then a dual f^* is described by the transpose M^T.

Theorem a) f^* computes the same values as f iff its matrix is symmetric.
b) In particular, it uses the same registers for output as input.

- In the normalised case, unused registers give columns of zeros.
- Used, non-output registers are over-written with 1_G: more zeros.
- Used, non-input registers are initialised to 1_G: more zeros.
- So only the sub-matrix M_{IO} on I/O registers need be symmetric.

Theorem A normalised space-aware chain for a *single* exponentiation and its dual compute the same values.

(Duals become unique only when written in atomic operations.)

Symmetric Cases

If the action of a (multi-) exponentiation function f on registers is described by matrix M then a dual f^* is described by the transpose M^T.

Theorem a) f^* computes the same values as f iff its matrix is symmetric.
b) In particular, it uses the same registers for output as input.

- In the normalised case, unused registers give columns of zeros.
- Used, non-output registers are over-written with 1_G: more zeros.
- Used, non-input registers are initialised to 1_G: more zeros.
- So only the sub-matrix M_{IO} on I/O registers need be symmetric.

Theorem A normalised space-aware chain for a *single* exponentiation and its dual compute the same values.

(Duals become unique only when written in atomic operations.)

New Algorithms

Information Security Group

High Level Algorithms

Question: When is an algorithm *dualisable* if its steps are more complex than the atomic operations?

We want to be able to decompose steps independently into atomic op^{ns} yet obtain the normalised property when all steps are concatenated.

Solution: For each step the values initially in its non-input registers must not be used and its used non-output registers must be reset to 1_G.

The output registers for one step must be the input registers for the next. (Include unused registers in the I/O set for convenience here.)

These are only requirements on how steps are realised as space-aware chains. So not a restriction on algorithm formulation.

Definition The dual of a high level exponentiation algorithm is that given by transposing its steps and reversing their order.

High Level Algorithms

Question: When is an algorithm *dualisable* if its steps are more complex than the atomic operations?

We want to be able to decompose steps independently into atomic op^{ns} yet obtain the normalised property when all steps are concatenated.

Solution: For each step the values initially in its non-input registers must not be used and its used non-output registers must be reset to 1_G.

The output registers for one step must be the input registers for the next. (Include unused registers in the I/O set for convenience here.)

These are only requirements on how steps are realised as space-aware chains. So not a restriction on algorithm formulation.

Definition The dual of a high level exponentiation algorithm is that given by transposing its steps and reversing their order.

High Level Algorithms

Question: When is an algorithm *dualisable* if its steps are more complex than the atomic operations?

We want to be able to decompose steps independently into atomic op^{ns} yet obtain the normalised property when all steps are concatenated.

Solution: For each step the values initially in its non-input registers must not be used and its used non-output registers must be reset to 1_G.

The output registers for one step must be the input registers for the next. (Include unused registers in the I/O set for convenience here.)

These are only requirements on how steps are realised as space-aware chains. So not a restriction on algorithm formulation.

Definition The dual of a high level exponentiation algorithm is that given by transposing its steps and reversing their order.

An "Old" Algorithm (Arith13, 1997)

Inputs: $g \in G, \quad D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$
Output: $g^D \in G$

$T \leftarrow 1_G$
$P \leftarrow g$
for $i \leftarrow 0$ **to** $n-1$ **do** $\{$
  if $d_i \neq 0$ **then** $T \leftarrow T \times P^{d_i}$
  if $i \neq n-1$ **then** $P \leftarrow P^{r_i}$ $\}$
return T

The loop body involves computing P^{d_i} *en route* to P^{r_i}.

Information Security Group

One Iteration

Base/digit pairs (r, d) are chosen for compact, fast performance.
Specifically at most one register in addition to P and T.

e.g. $r = 2^i \pm 1, d = 2^j$ will involve i squarings & 2 mult^s.

It avoids a table entry for each d.

There is now a dual algorithm using the same space – only three registers.

The step $T \leftarrow TP^d$, $P \leftarrow P^r$ is achieved by $\begin{bmatrix} r & 0 \\ d & 1 \end{bmatrix} = \begin{bmatrix} r & d \\ 0 & 1 \end{bmatrix}^{\mathsf{T}}$.

So the transpose performs the dual op^n $P \leftarrow P^r T^d$.

The sequence of op^s is easily determined via the dual.

One Iteration

Base/digit pairs (r, d) are chosen for compact, fast performance.
Specifically at most one register in addition to P and T.

e.g. $r = 2^i \pm 1, d = 2^j$ will involve i squarings & 2 mult^s.

It avoids a table entry for each d.

There is now a dual algorithm using the same space – only three registers.

The step $T \leftarrow TP^d$, $P \leftarrow P^r$ is achieved by $\begin{bmatrix} r & 0 \\ d & 1 \end{bmatrix} = \begin{bmatrix} r & d \\ 0 & 1 \end{bmatrix}^{\mathsf{T}}$.

So the transpose performs the dual op^n $P \leftarrow P^r T^d$.

The sequence of op^s is easily determined via the dual.

A New Compact Left-to-Right Algorithm

Inputs: $g \in G, \quad D = ((d_{n-1}r_{n-2}+d_{n-2})r_{n-3}+\ldots+d_1)r_0+d_0 \in \mathbb{N}$
Output: $g^D \in G$

$T \leftarrow g$
$P \leftarrow 1_G$
for $i \leftarrow n-1$ **downto** 0 **do**
    $P \leftarrow P^{r_i} \times T^{d_i}$
return P

Loop iterations are computed as described on last slide.

It is the dual of the previous R2L algorithm, as just derived.

Information Security Group

The Value of the Algorithm

- "Table-less" exponentiation – useful in constrained environments.
- If space for only three registers and division has the same cost as mult$^{\mathsf{n}}$, the compact algorithms are faster.
- A left-to-right version allows better use of composite op$^{\mathsf{s}}$, e.g. double-and-add, triple-and-add, quintuple-and-add.
- Recoding is done on-the-fly for R2L exp$^{\mathrm{n}}$; in advance for L2R exp$^{\mathrm{n}}$. The recoding typically needs up to 3 times the storage space of D.

Information Security Group

Conclusion

Information Security Group

Summary & Final Remarks

- A general setting enabling most exp^n algorithms to be described naturally, namely a mixed base recoding.
- A new space- and time-preserving duality between left-to-right and right-to-left exp^n algorithms.
- A new tableless exp^n algorithm.
  It enables new speed records to be set in certain environments.
- New understanding of exp^n is possible,
  e.g. a comparison of R2L initialisation with L2R finalisation steps.
- . . .

Information Security Group

Summary & Final Remarks

- A general setting enabling most $\exp^{\mathrm{n}}$ algorithms to be described naturally, namely a mixed base recoding.
- A new space- and time-preserving duality between left-to-right and right-to-left $\exp^{\mathrm{n}}$ algorithms.
- A new tableless $\exp^{n}$ algorithm. It enables new speed records to be set in certain environments.
- New understanding of $\exp^{\mathrm{n}}$ is possible, e.g. a comparison of R2L initialisation with L2R finalisation steps.
- . . .

Information Security Group

Summary & Final Remarks

- A general setting enabling most $\exp^{\mathrm{n}}$ algorithms to be described naturally, namely a mixed base recoding.
- A new space- and time-preserving duality between left-to-right and right-to-left $\exp^{\mathrm{n}}$ algorithms.
- A new tableless $\exp^{n}$ algorithm. It enables new speed records to be set in certain environments.
- New understanding of $\exp^{\mathrm{n}}$ is possible, e.g. a comparison of R2L initialisation with L2R finalisation steps.
- . . .

Information Security Group

Summary & Final Remarks

- A general setting enabling most $\exp^{n}$ algorithms to be described naturally, namely a mixed base recoding.
- A new space- and time-preserving duality between left-to-right and right-to-left $\exp^{n}$ algorithms.
- A new tableless $\exp^{n}$ algorithm.
  It enables new speed records to be set in certain environments.
- New understanding of $\exp^{n}$ is possible,
  e.g. a comparison of R2L initialisation with L2R finalisation steps.
- ...

Information Security Group

Summary & Final Remarks

- A general setting enabling most exp^{n} algorithms to be described naturally, namely a mixed base recoding.
- A new space- and time-preserving duality between left-to-right and right-to-left exp^{n} algorithms.
- A new tableless $\exp^n$ algorithm. It enables new speed records to be set in certain environments.
- New understanding of exp^{n} is possible, e.g. a comparison of R2L initialisation with L2R finalisation steps.
- ...

Information Security Group

CT-RSA 2012 — February 29th, 2012

Optimal Eta Pairing on Supersingular Genus-2 Binary Hyperelliptic Curves

Nicolas Estibals

CARAMEL project-team, LORIA,
Université de Lorraine / CNRS / INRIA, France
Nicolas.Estibals@loria.fr

Joint work with:

Diego F. Aranha	Institute of Computing, University of Campinas, Brazil
Jean-Luc Beuchat	Graduate School of Systems and Information Engineering, University of Tsukuba, Japan
Jérémie Detrey	CARAMEL project-team, LORIA, INRIA / Université de Lorraine / CNRS, France

Pairings and cryptology

- used as a primitive in many protocols and devices
  - Boneh–Lynn–Shacham short signature
  - Boneh–Franklin identity-based encryption
  - . . .

Pairings and cryptology

- used as a primitive in many protocols and devices
  - Boneh–Lynn–Shacham short signature
  - Boneh–Franklin identity-based encryption
  - ...
- implementations needed for various targets
  - online server $\rightarrow$ high-speed software
  - smart card $\rightarrow$ low-resource hardware
- reach 128 bits of security (equivalent to AES)

What's a cryptographic pairing

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

- where $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \times)$ are cyclic groups of order ℓ
- The discrete logarithm problem should be hard on these groups

What's a cryptographic pairing

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

- where $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \times)$ are cyclic groups of order ℓ
- The discrete logarithm problem should be hard on these groups
- Bilinear map:
$$e(aP, bQ) = e(P, Q)^{ab}$$

What's a cryptographic pairing

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

- where $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \times)$ are cyclic groups of order ℓ
- The discrete logarithm problem should be hard on these groups
- Bilinear map:
$$e(aP, bQ) = e(P, Q)^{ab}$$
- Symmetric pairing (Type-1): $\mathbb{G}_1 = \mathbb{G}_2$, exploited by some protocols

What's a cryptographic pairing

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

- where $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$ and $(\mathbb{G}_T, \times)$ are cyclic groups of order ℓ
- The discrete logarithm problem should be hard on these groups
- Bilinear map:
$$e(aP, bQ) = e(P, Q)^{ab}$$
- Symmetric pairing (Type-1): $\mathbb{G}_1 = \mathbb{G}_2$, exploited by some protocols
- Choice of the groups:
  - $\mathbb{G}_1$, $\mathbb{G}_2$: related to an algebraic curve
  - $\mathbb{G}_T$: related to the field of definition of the curve

Classical choice of curves

Barreto–Naehrig curves

+ Lots of literature
+ Huge optimization efforts
+ Suited for 128 bits of security

– Arithmetic modulo $p \approx 256$ bits
– No symmetric pairing

Classical choice of curves

Barreto–Naehrig curves

- \+ Lots of literature
- \+ Huge optimization efforts
- \+ Suited for 128 bits of security
- – Arithmetic modulo $p \approx 256$ bits
- – No symmetric pairing

Supersingular elliptic curves

- \+ Symmetric pairing

  Thanks to a distortion map

  $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$

- \+ Small characteristic arithmetic

  $\Rightarrow$ No carry propagation

- – Not suited to 128-bit security level

  Larger base field: $\mathbb{F}_{2^{1223}}$, $\mathbb{F}_{3^{509}}$

Classical choice of curves

Barreto–Naehrig curves

+ Lots of literature
+ Huge optimization efforts
+ Suited for 128 bits of security
– Arithmetic modulo $p \approx 256$ bits
– No symmetric pairing

Supersingular elliptic curves

+ Symmetric pairing
  Thanks to a distortion map
  $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$
+ Small characteristic arithmetic
  $\Rightarrow$ No carry propagation
– Not suited to 128-bit security level
  Larger base field: $\mathbb{F}_{2^{1223}}$, $\mathbb{F}_{3^{509}}$

Classical choice of curves

Barreto–Naehrig curves

+ Lots of literature
+ Huge optimization efforts
+ Suited for 128 bits of security
– Arithmetic modulo $p \approx 256$ bits
– No symmetric pairing

Supersingular elliptic curves

+ Symmetric pairing
  Thanks to a distortion map
  $\psi : \mathbb{G}_1 \rightarrow \mathbb{G}_2$
+ Small characteristic arithmetic
  $\Rightarrow$ No carry propagation
– Not suited to 128-bit security level
  Larger base field: $\mathbb{F}_{2^{1223}}$, $\mathbb{F}_{3^{509}}$

- Solutions to the large base field needed by supersingular curves
  - (Pairing 2010) Use fields of composite extension degree: benefit from faster field arithmetic but requires careful security analysis

Classical choice of curves

Barreto–Naehrig curves

+ Lots of literature
+ Huge optimization efforts
+ Suited for 128 bits of security
– Arithmetic modulo $p \approx 256$ bits
– No symmetric pairing

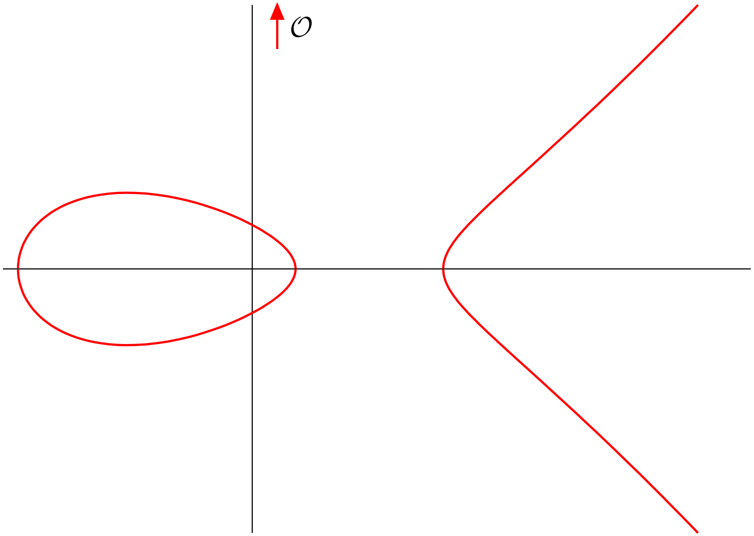
Supersingular elliptic curves

+ Symmetric pairing
  Thanks to a distortion map
  $\psi : \mathbb{G}_1 \to \mathbb{G}_2$
+ Small characteristic arithmetic
  $\Rightarrow$ No carry propagation
– Not suited to 128-bit security level
  Larger base field: $\mathbb{F}_{2^{1223}}$, $\mathbb{F}_{3^{509}}$

- Solutions to the large base field needed by supersingular curves
  - (Pairing 2010) Use fields of composite extension degree: benefit from faster field arithmetic but requires careful security analysis
  - (This work) Use genus-2 hyperelliptic curves: base field will be $\mathbb{F}_{2^{367}}$

Elliptic curves

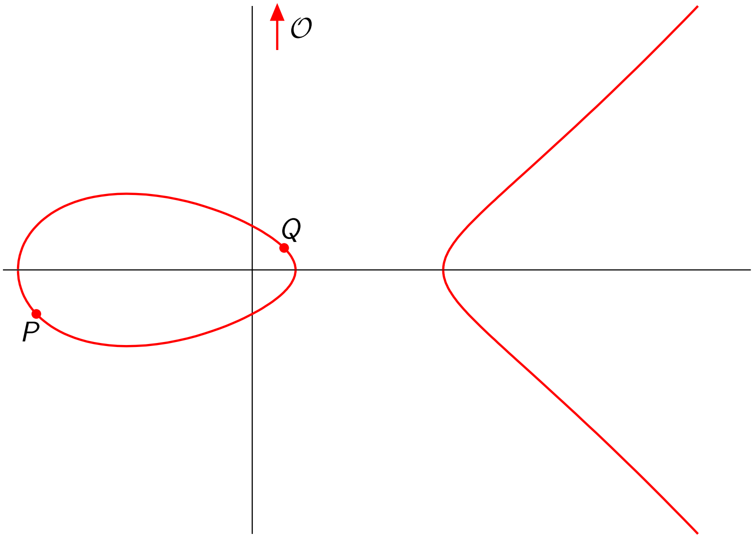
$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Elliptic curves

- $E(K)$ is a group

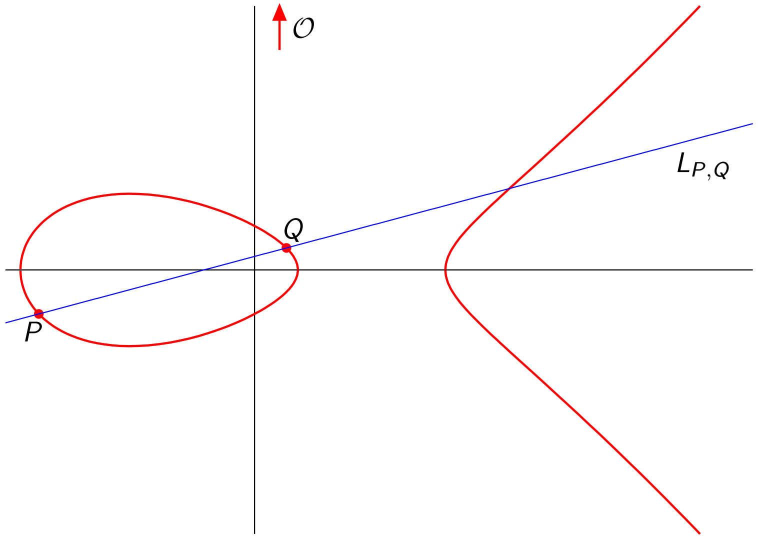
$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Elliptic curves

- $E(K)$ is a group

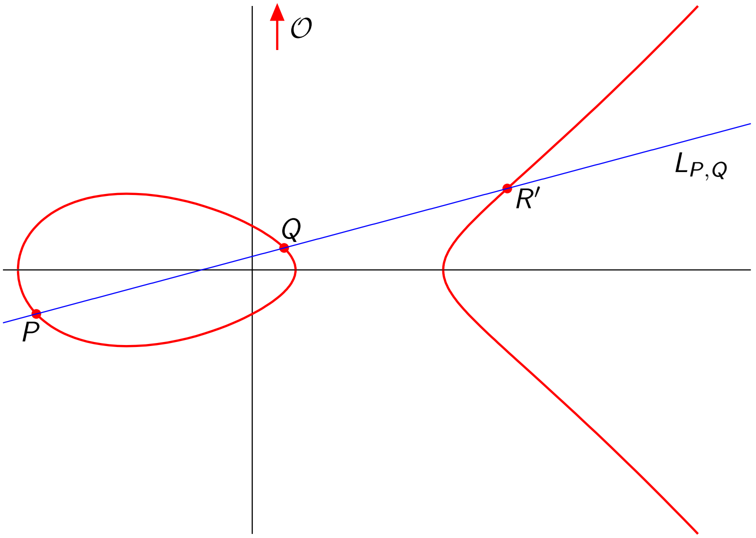
$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Elliptic curves

- $E(K)$ is a group

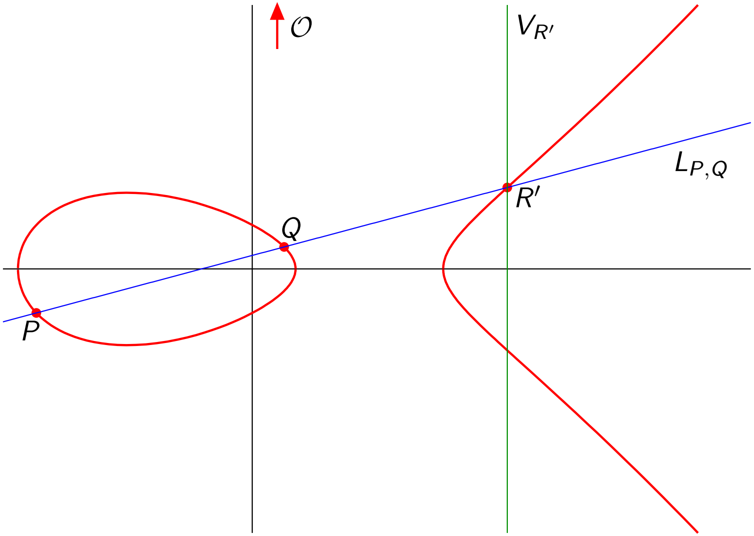
$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Elliptic curves

- $E(K)$ is a group

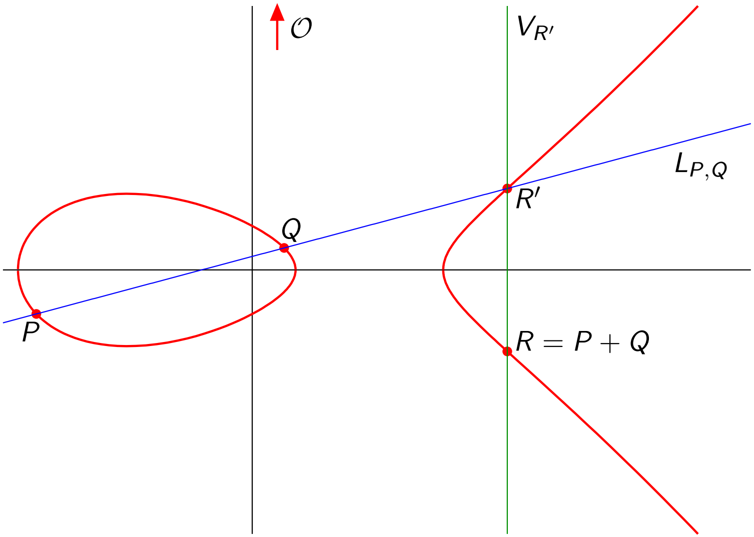
$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Elliptic curves

- $E(K)$ is a group

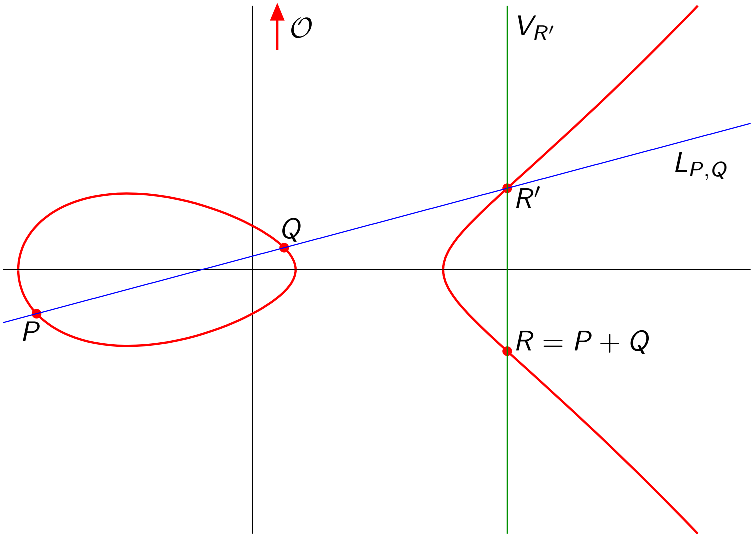
$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Elliptic curves

$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$

- $E(K)$ is a group
- In practice: K is a finite field $\mathbb{F}_q$
- $E(\mathbb{F}_q)$ is a finite group

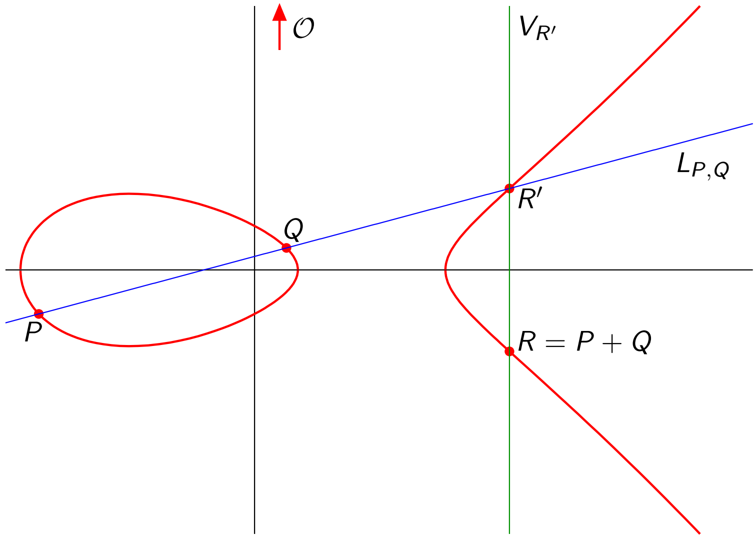


Elliptic curves

- $E(K)$ is a group
- In practice: K is a finite field $\mathbb{F}_q$
- $E(\mathbb{F}_q)$ is a finite group
- ℓ: a large prime dividing $\#E(\mathbb{F}_q)$
- Use the cyclic subgroup

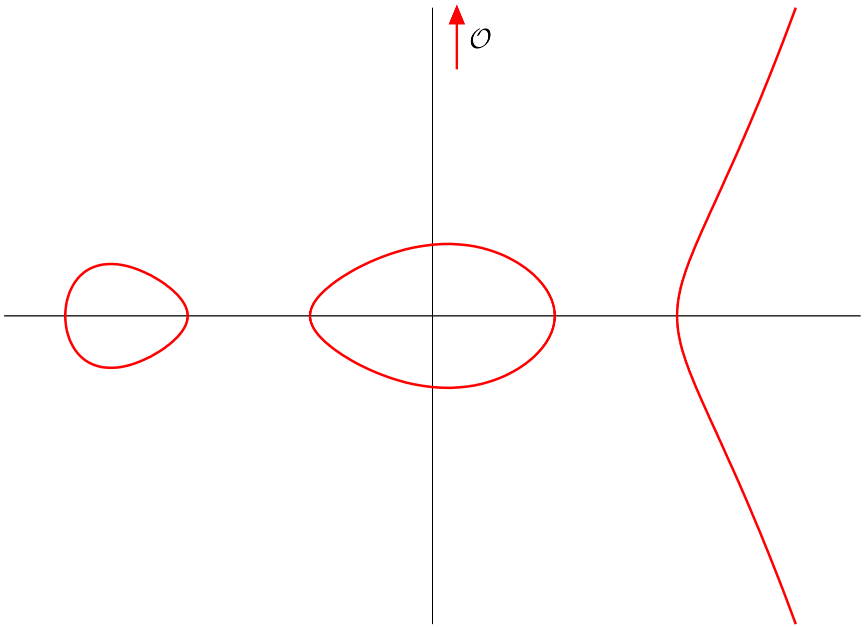
$$E(\mathbb{F}_q)[\ell] = \{P \mid [\ell]P = \mathcal{O}\}$$

$$E/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 1$ and $\deg f = 3$



Genus-2 hyperelliptic curves

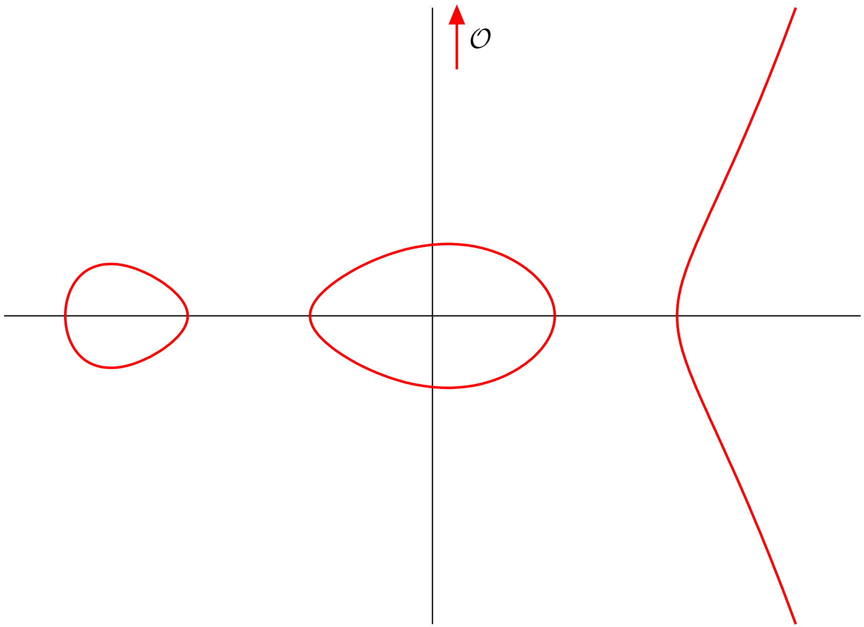
$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$



Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

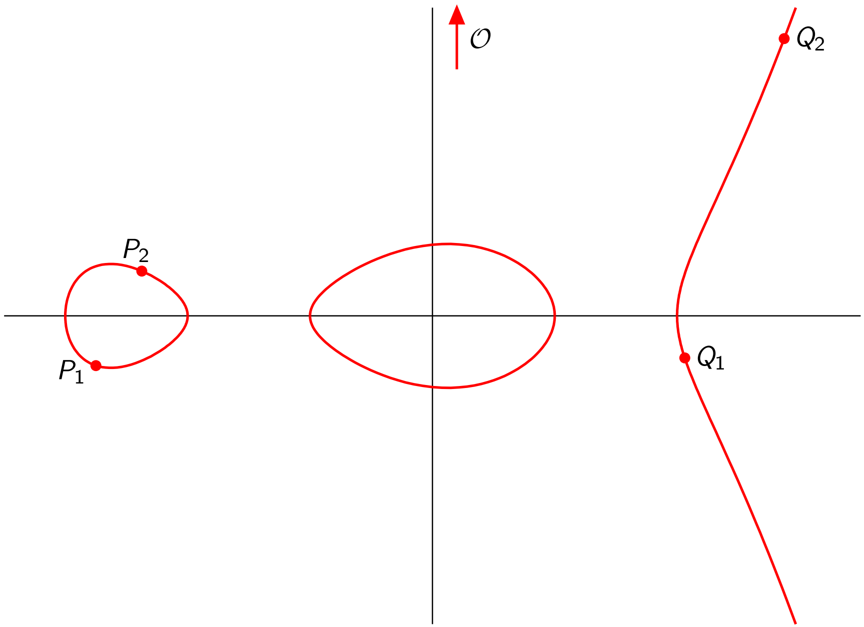
- $C(K)$ not a group!



Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

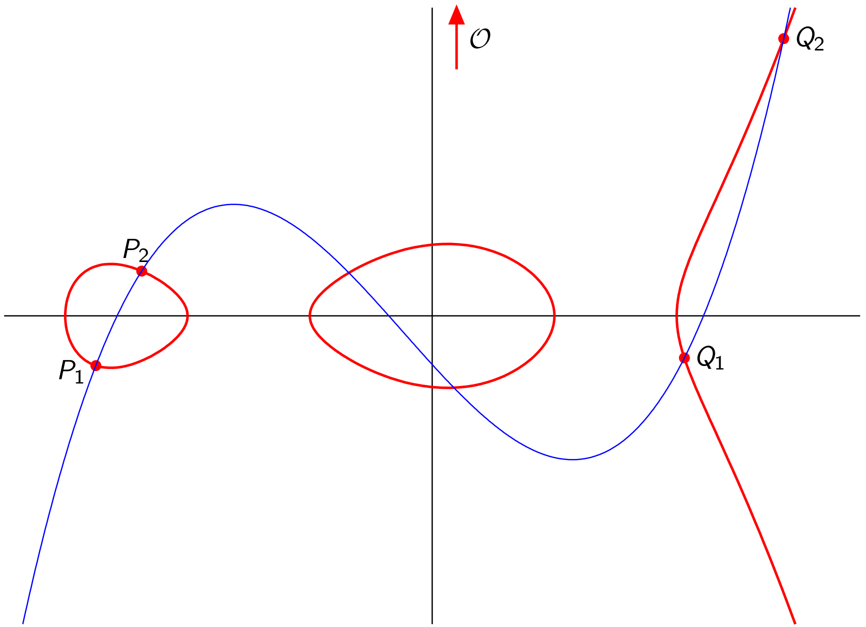
- $C(K)$ not a group!
- But pairs of points $\{P_1, P_2\}$



Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

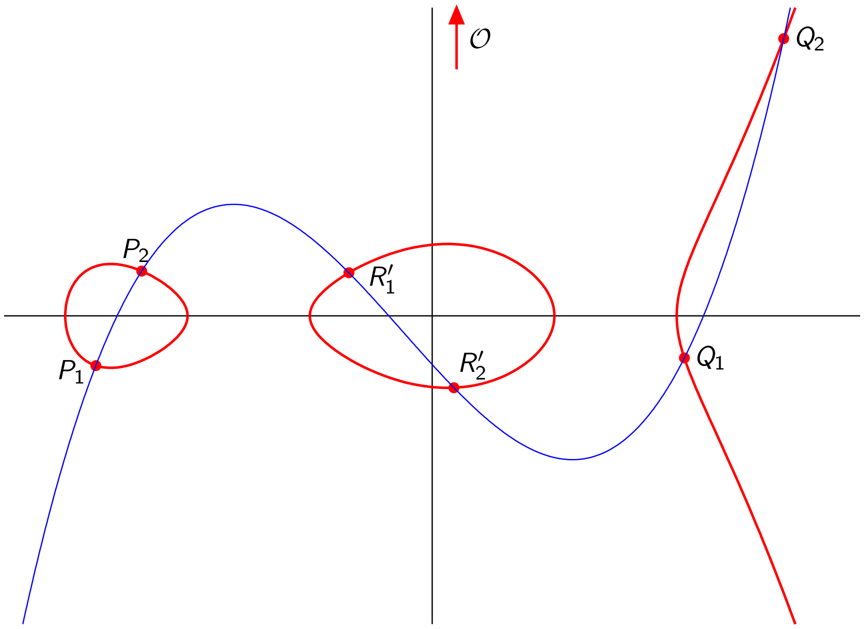
- $C(K)$ not a group!
- But pairs of points
  $\{P_1, P_2\}$



Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

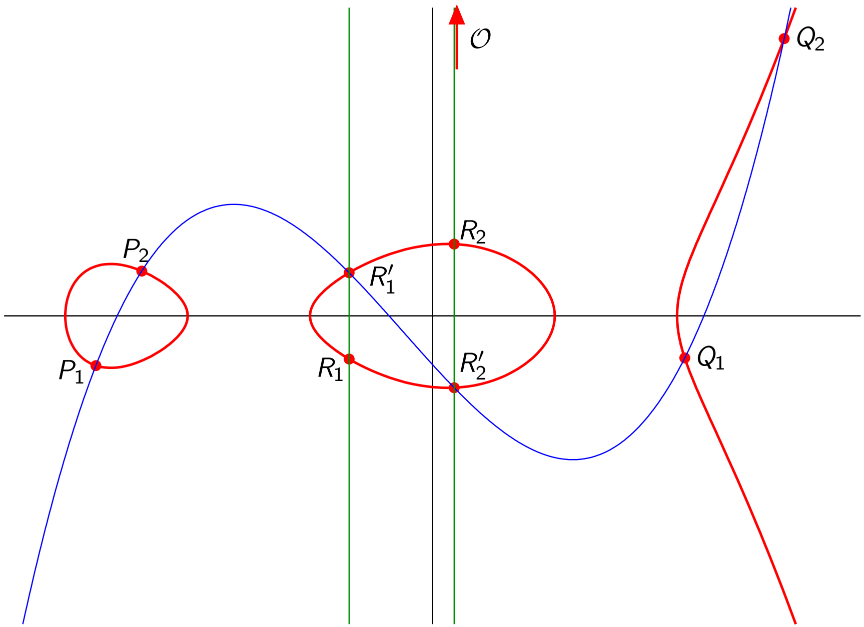
- $C(K)$ not a group!
- But pairs of points $\{P_1, P_2\}$



Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with deg $h \leq 2$ and deg $f = 5$

- $C(K)$ not a group!
- But pairs of points $\{P_1, P_2\}$

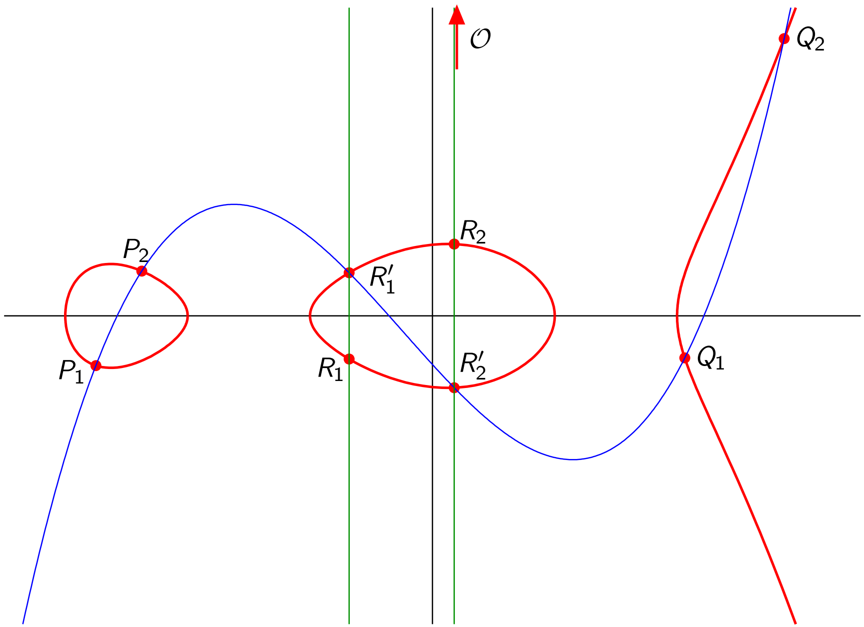


$$\{P_1, P_2\} + \{Q_1, Q_2\} = \{R_1, R_2\}$$

Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

- $C(K)$ not a group!
- But pairs of points
  $\{P_1, P_2\}$
- More formally
  - use the Jacobian
    $\mathrm{Jac}_C(K)$

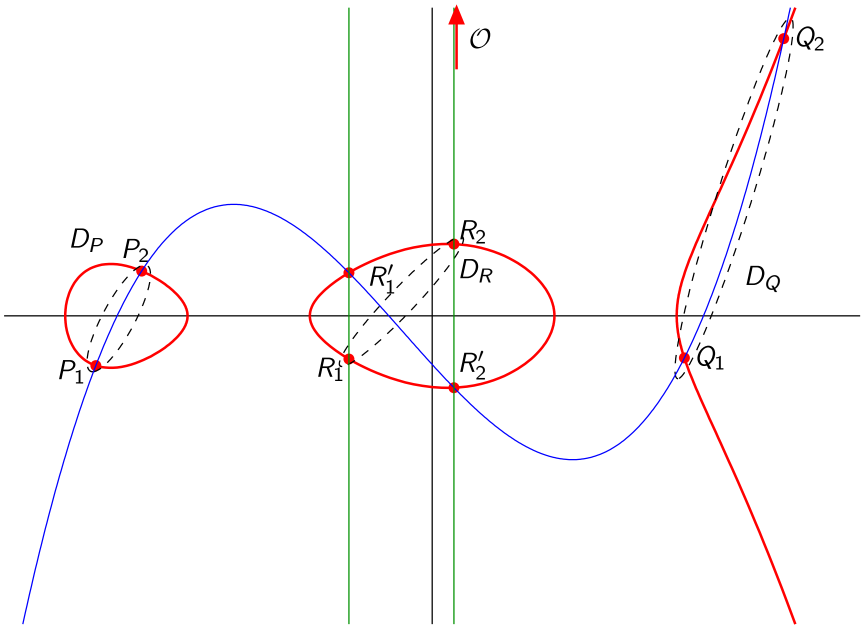


$$\{P_1, P_2\} + \{Q_1, Q_2\} = \{R_1, R_2\}$$

Genus-2 hyperelliptic curves

- $C(K)$ not a group!
- But pairs of points
  $$\{P_1, P_2\}$$
- More formally
  - use the Jacobian
    $$\mathrm{Jac}_C(K)$$

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

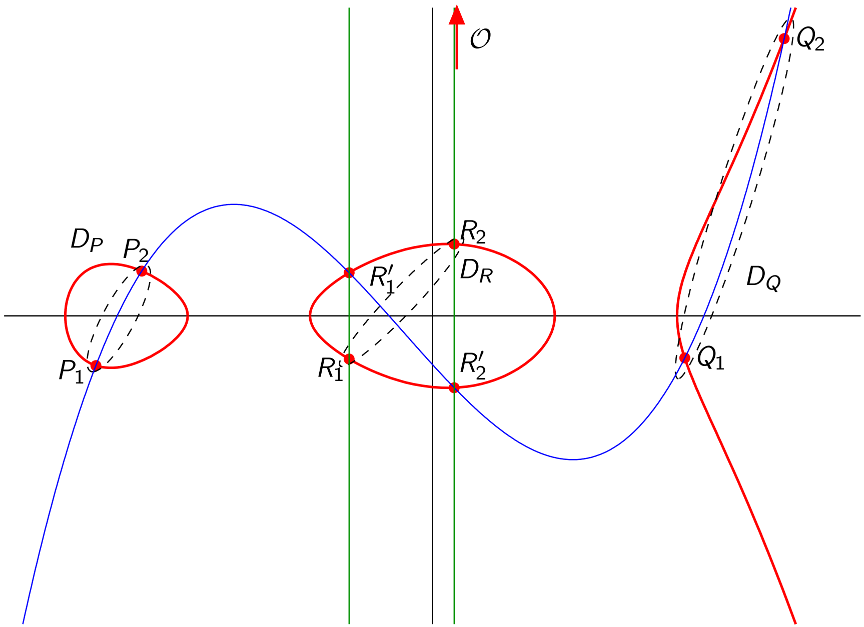


$$D_P + D_Q \qquad = D_R$$

Genus-2 hyperelliptic curves

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

- $C(K)$ not a group!
- But pairs of points
  $\{P_1, P_2\}$
- More formally
  - use the Jacobian
    $\mathrm{Jac}_C(K)$
  - general form of the elements (called divisor)
    $D_P = (P_1) + (P_2) - 2(\mathcal{O})$

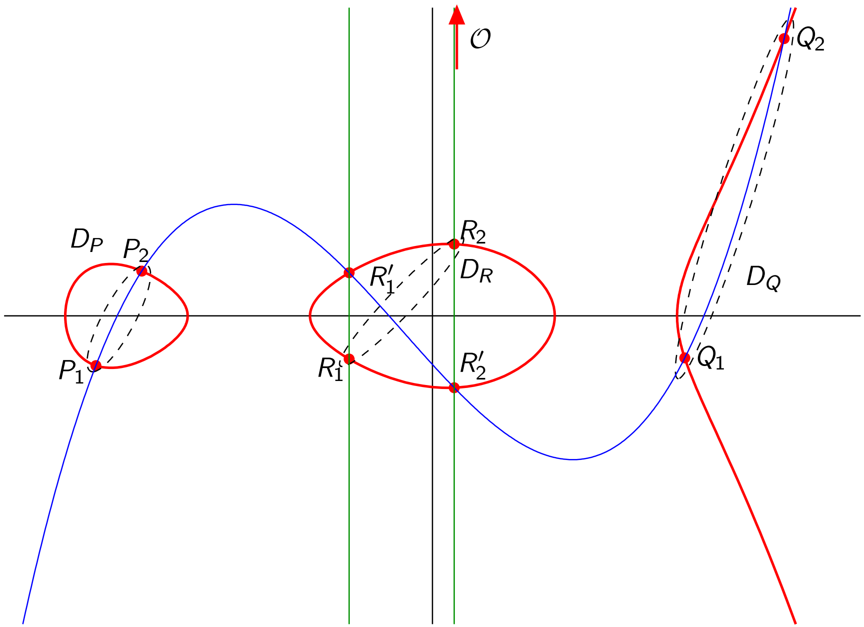


$$D_P + D_Q \qquad = D_R$$

Genus-2 hyperelliptic curves

- $C(K)$ not a group!
- But pairs of points
  $$\{P_1, P_2\}$$
- More formally
  - use the Jacobian
    $$\mathrm{Jac}_C(K)$$
  - general form of the elements (called divisor)
    $$D_P = (P_1) + (P_2) - 2(\mathcal{O})$$
  - degenerate form
    $$(P) - (\mathcal{O})$$

$$C/K : y^2 + h(x) \cdot y = f(x)$$
with $\deg h \leq 2$ and $\deg f = 5$

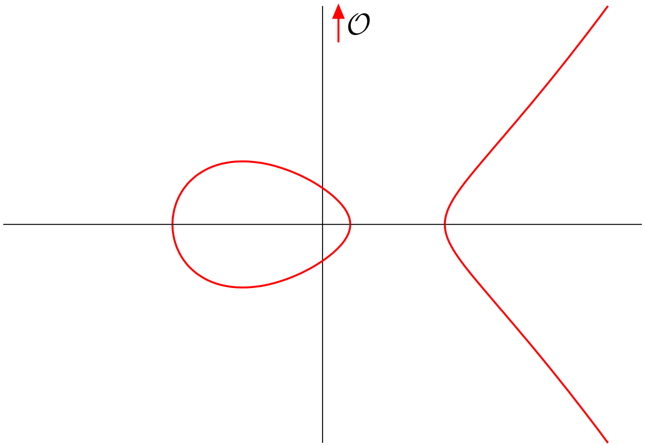


$$D_P + D_Q \qquad = D_R$$

Computing the pairing: Miller's algorithm (elliptic case)

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

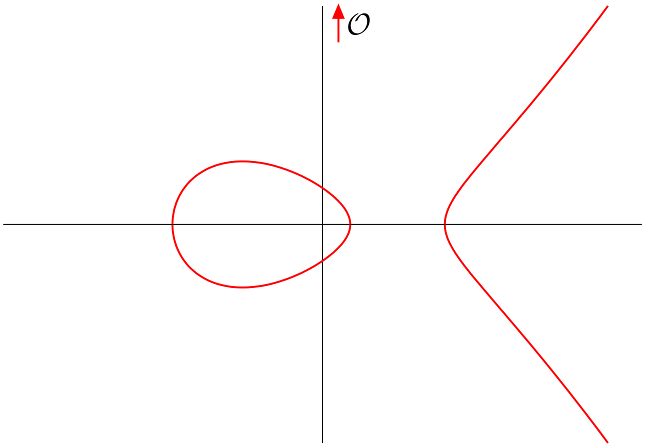
- Reduced Tate pairing



Computing the pairing: Miller's algorithm (elliptic case)

$$\begin{array}{rcccl} e: & E(\mathbb{F}_q)[\ell] & \times & \mathbb{G}_2 & \longrightarrow \quad \mathbb{G}_T \\ & P & , & & \end{array}$$

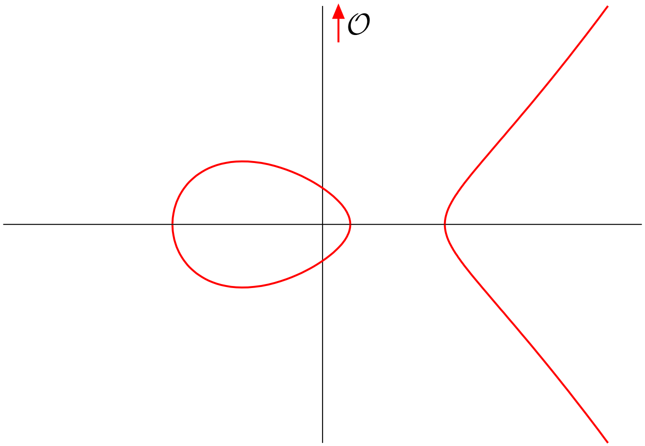
- Reduced Tate pairing



Computing the pairing: Miller's algorithm (elliptic case)

$$\begin{array}{rcccc} e: & E(\mathbb{F}_q)[\ell] & \times & E(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mathbb{G}_T \\ & P & , & Q & \end{array}$$

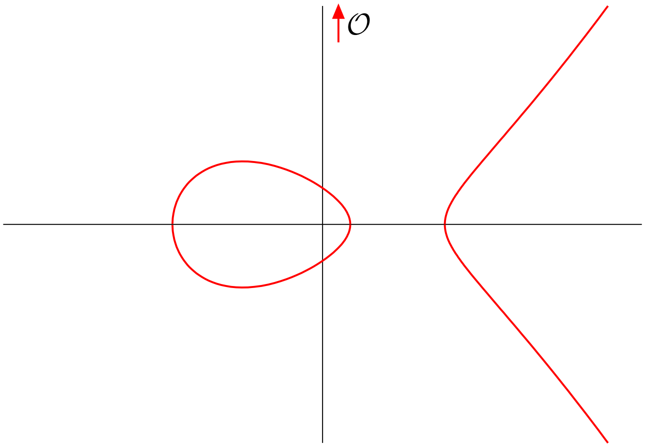
- Reduced Tate pairing
- k: embedding degree (curve parameter)



Computing the pairing: Miller's algorithm (elliptic case)

$$\begin{array}{rcccccl} e: & E(\mathbb{F}_q)[\ell] & \times & E(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & P & , & Q & \longmapsto & f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \end{array}$$

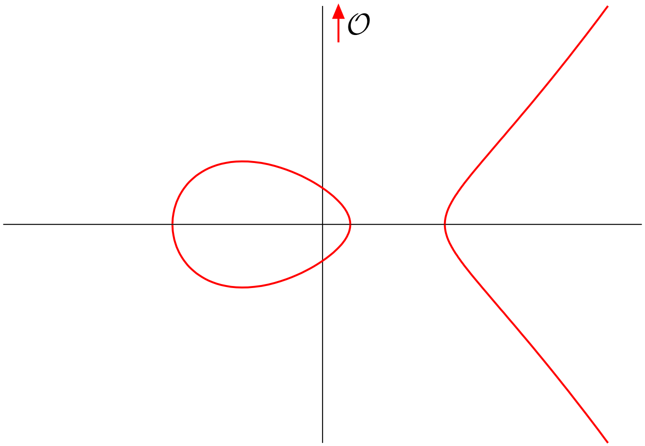
- Reduced Tate pairing
- k: embedding degree (curve parameter)



Computing the pairing: Miller's algorithm (elliptic case)

$$\begin{array}{rccccc} e: & E(\mathbb{F}_q)[\ell] & \times & E(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & P & , & Q & \longmapsto & f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \end{array}$$

- Reduced Tate pairing
- k: embedding degree (curve parameter)
- Miller functions: $f_{n,P}$

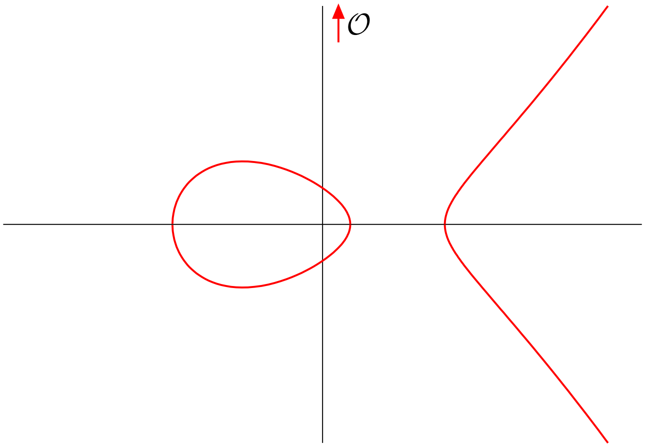


Computing the pairing: Miller's algorithm (elliptic case)

$$\begin{array}{rccccl} e: & E(\mathbb{F}_q)[\ell] & \times & E(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & P & , & Q & \longmapsto & f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \end{array}$$

- Reduced Tate pairing
- k: embedding degree (curve parameter)
- Miller functions: $f_{n,P}$
  - an inductive identity

$$\begin{array}{rcl} f_{1,P} & = & 1 \\ f_{n+n',P} & = & f_{n,P} \cdot f_{n',P} \cdot g_{[n]P,[n']P} \end{array}$$



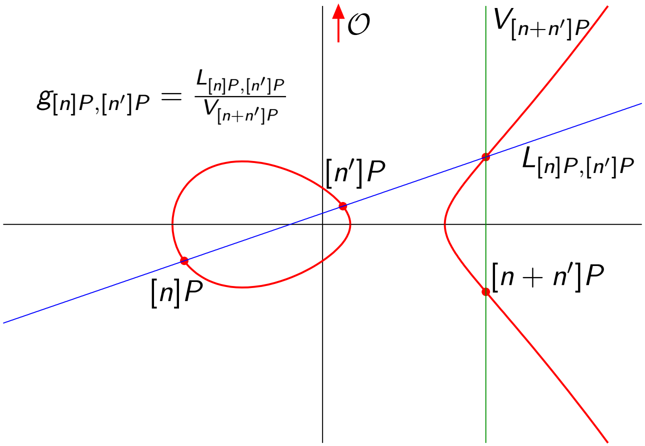
Computing the pairing: Miller's algorithm (elliptic case)

$$\begin{array}{rcccl} e: & E(\mathbb{F}_q)[\ell] & \times & E(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & P & , & Q & \longmapsto & f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \end{array}$$

- Reduced Tate pairing
- k: embedding degree (curve parameter)
- Miller functions: $f_{n,P}$
  - an inductive identity

$$\begin{array}{rcl} f_{1,P} & = & 1 \\ f_{n+n',P} & = & f_{n,P} \cdot f_{n',P} \cdot g_{[n]P,[n']P} \end{array}$$

  - $g_{[n]P,[n']P}$ derived from the addition of $[n]P$ and $[n']P$



Computing the pairing: Miller's algorithm (elliptic case)

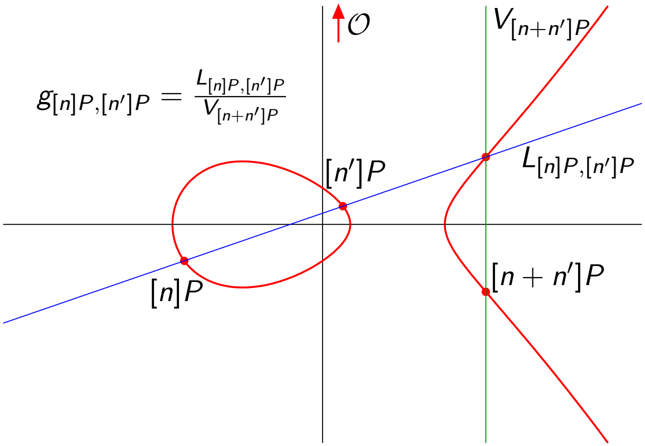
$$\begin{array}{rcccl} e: & E(\mathbb{F}_q)[\ell] & \times & E(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & P & , & Q & \longmapsto & f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \end{array}$$

- Reduced Tate pairing
- k: embedding degree (curve parameter)
- Miller functions: $f_{n,P}$
  - an inductive identity

$$\begin{array}{rcl} f_{1,P} & = & 1 \\ f_{n+n',P} & = & f_{n,P} \cdot f_{n',P} \cdot g_{[n]P,[n']P} \end{array}$$

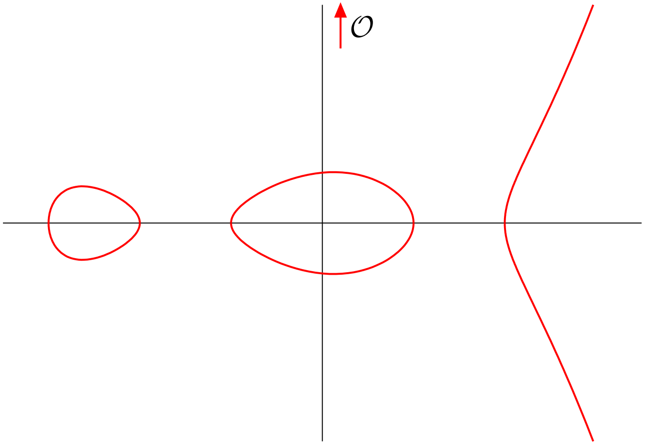
  - $g_{[n]P,[n']P}$ derived from the addition of $[n]P$ and $[n']P$
  - compute $f_{\ell,P}$ thanks to an addition chain
  - in practice: double-and-add

$\log_2 \ell$ iterations



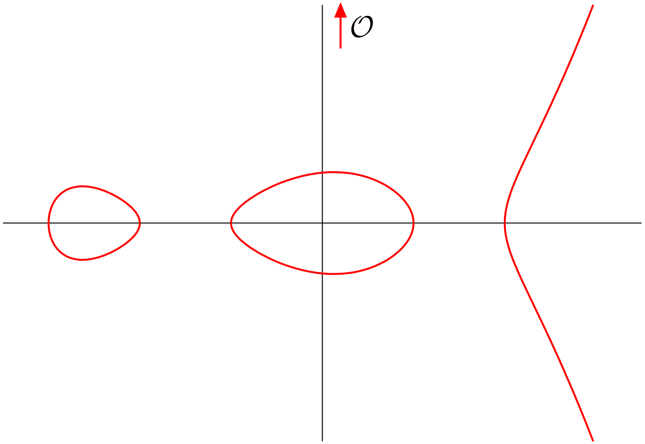
Miller's algorithm (hyperelliptic case)

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



Miller's algorithm (hyperelliptic case)

$$e: \quad \mathrm{Jac}_C(\mathbb{F}_q)[\ell] \;\times\; \mathrm{Jac}_C(\mathbb{F}_{q^k})[\ell] \;\longrightarrow\; \mu_\ell \subset \mathbb{F}_{q^k}^*$$

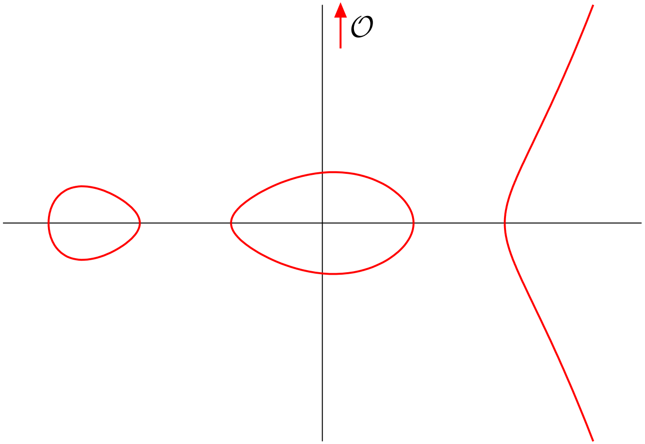


Miller's algorithm (hyperelliptic case)

$$\begin{array}{rcccl} e: & \mathrm{Jac}_C(\mathbb{F}_q)[\ell] & \times & \mathrm{Jac}_C(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & D_1 & , & D_2 & \longmapsto & f_{\ell,D_1}(D_2)^{\frac{q^k-1}{\ell}} \end{array}$$

- Hyperelliptic Miller functions: $f_{n,D}$
  - same inductive identity

$$\begin{aligned} f_{1,D} &= 1 \\ f_{n+n',D} &= f_{n,D} \cdot f_{n',D} \cdot g_{[n]D,[n']D} \end{aligned}$$



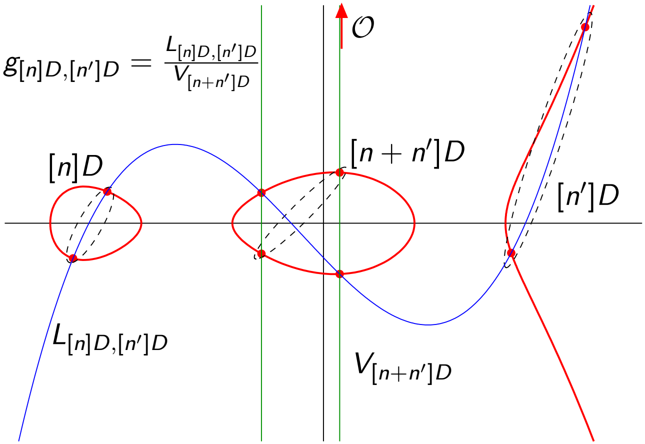
Miller's algorithm (hyperelliptic case)

$$\begin{array}{rcccl} e: & \mathrm{Jac}_C(\mathbb{F}_q)[\ell] & \times & \mathrm{Jac}_C(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & D_1 & , & D_2 & \longmapsto & f_{\ell,D_1}(D_2)^{\frac{q^k-1}{\ell}} \end{array}$$

- Hyperelliptic Miller functions: $f_{n,D}$
  - same inductive identity

$$\begin{aligned} f_{1,D} &= 1 \\ f_{n+n',D} &= f_{n,D} \cdot f_{n',D} \cdot g_{[n]D,[n']D} \end{aligned}$$

  - $g_{[n]D,[n']D}$ derived from the addition of $[n]D$ and $[n']D$
  - use Cantor's addition algorithm



Miller's algorithm (hyperelliptic case)

$$\begin{array}{rccccl} e: & \mathrm{Jac}_C(\mathbb{F}_q)[\ell] & \times & \mathrm{Jac}_C(\mathbb{F}_{q^k})[\ell] & \longrightarrow & \mu_\ell \subset \mathbb{F}_{q^k}^* \\ & D_1 & , & D_2 & \longmapsto & f_{\ell,D_1}(D_2)^{\frac{q^k-1}{\ell}} \end{array}$$

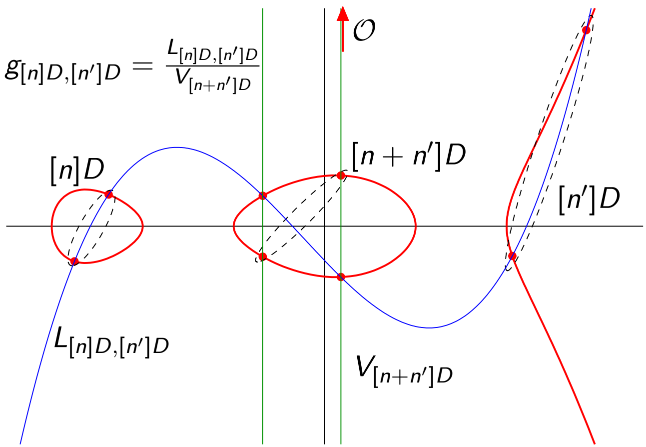
- Hyperelliptic Miller functions: $f_{n,D}$
  - same inductive identity

$$\begin{aligned} f_{1,D} &= 1 \\ f_{n+n',D} &= f_{n,D} \cdot f_{n',D} \cdot g_{[n]D,[n']D} \end{aligned}$$

  - $g_{[n]D,[n']D}$ derived from the addition of $[n]D$ and $[n']D$
  - use Cantor's addition algorithm
  - double-and-add algorithm

    $\log_2 \ell$ iterations

  - iterations are more complex



Genus-2 binary supersingular curve: our choice

$$C_d/\mathbb{F}_{2^m} : y^2 + y = x^5 + x^3 + d \text{ with } d \in \mathbb{F}_2$$

- A distortion map exists: symmetric pairing
- $\# \operatorname{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$
- Embedding degree of the curve: $k = 12$
- For 128 bits of security: $\mathbb{F}_{2^m} = \mathbb{F}_{2^{367}}$ and $d = 0$

Genus-2 binary supersingular curve: our choice

$$C_d/\mathbb{F}_{2^m} : y^2 + y = x^5 + x^3 + d \text{ with } d \in \mathbb{F}_2$$

- A distortion map exists: symmetric pairing
- $\# \operatorname{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$
- Embedding degree of the curve: $k = 12$
- For 128 bits of security: $\mathbb{F}_{2^m} = \mathbb{F}_{2^{367}}$ and $d = 0$
- Key property of the curve:

$$[2]\,((P) - (\mathcal{O})) \;=\; (P) + (P) - 2(\mathcal{O})$$

Genus-2 binary supersingular curve: our choice

$$C_d/\mathbb{F}_{2^m} : y^2 + y = x^5 + x^3 + d \text{ with } d \in \mathbb{F}_2$$

- A distortion map exists: symmetric pairing
- $\# \operatorname{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$
- Embedding degree of the curve: $k = 12$
- For 128 bits of security: $\mathbb{F}_{2^m} = \mathbb{F}_{2^{367}}$ and $d = 0$
- Key property of the curve:

$$\begin{aligned} [2]\,((P) - (\mathcal{O})) &= (P) + (P) - 2(\mathcal{O}) \\ [4]\,((P) - (\mathcal{O})) &= (P_4) + (P_4') - 2(\mathcal{O}) \end{aligned}$$

Genus-2 binary supersingular curve: our choice

$$C_d/\mathbb{F}_{2^m} : y^2 + y = x^5 + x^3 + d \text{ with } d \in \mathbb{F}_2$$

- A distortion map exists: symmetric pairing
- $\# \operatorname{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$
- Embedding degree of the curve: $k = 12$
- For 128 bits of security: $\mathbb{F}_{2^m} = \mathbb{F}_{2^{367}}$ and $d = 0$
- Key property of the curve:

$$\begin{aligned}
[2]\,((P) - (\mathcal{O})) &= (P) + (P) - 2(\mathcal{O}) \\
[4]\,((P) - (\mathcal{O})) &= (P_4) + (P_4') - 2(\mathcal{O}) \\
[8]\,((P) - (\mathcal{O})) &= (P_8) \quad - (\mathcal{O})
\end{aligned}$$

Genus-2 binary supersingular curve: our choice

$$C_d/\mathbb{F}_{2^m} : y^2 + y = x^5 + x^3 + d \text{ with } d \in \mathbb{F}_2$$

- A distortion map exists: symmetric pairing
- $\# \operatorname{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$
- Embedding degree of the curve: $k = 12$
- For 128 bits of security: $\mathbb{F}_{2^m} = \mathbb{F}_{2^{367}}$ and $d = 0$
- Key property of the curve:

$$\begin{aligned}
[2]\left((P) - (\mathcal{O})\right) &= (P) + (P) - 2(\mathcal{O}) \\
[4]\left((P) - (\mathcal{O})\right) &= (P_4) + (P_4') - 2(\mathcal{O}) \\
[8]\left((P) - (\mathcal{O})\right) &= ([8]P) - (\mathcal{O})
\end{aligned}$$

  - octupling acts on the curve

Genus-2 binary supersingular curve: our choice

$$C_d/\mathbb{F}_{2^m} : y^2 + y = x^5 + x^3 + d \text{ with } d \in \mathbb{F}_2$$

- A distortion map exists: symmetric pairing
- $\# \mathrm{Jac}_{C_d}(\mathbb{F}_{2^m}) = 2^{2m} \pm 2^{(3m+1)/2} + 2^m \pm 2^{(m+1)/2} + 1$
- Embedding degree of the curve: $k = 12$
- For 128 bits of security: $\mathbb{F}_{2^m} = \mathbb{F}_{2^{367}}$ and $d = 0$
- Key property of the curve:

$$\begin{aligned}
[2]\,((P) - (\mathcal{O})) &= (P) + (P) - 2(\mathcal{O}) \\
[4]\,((P) - (\mathcal{O})) &= (P_4) + (P_4') - 2(\mathcal{O}) \\
[8]\,((P) - (\mathcal{O})) &= ([8]P) - (\mathcal{O})
\end{aligned}$$

  - octupling acts on the curve
  - $f_{8,D}$ has a much simpler expression than $f_{2,D}$

Constructing the Optimal Eta pairing

Algorithm	Tate double & add			
#iterations	$2m$			

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \operatorname{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings

Constructing the Optimal Eta pairing

Algorithm	Tate double & add	Tate octuple & add		
#iterations	$2m$	$\frac{2m}{3}$		

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \mathrm{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings
- Use of octupling: simpler iteration also!

Constructing the Optimal Eta pairing

Algorithm	Tate double & add	Tate octuple & add	Barreto *et al.* η_T pairing	
#iterations	$2m$	$\frac{2m}{3}$	$\frac{m}{2}$	

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \operatorname{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings
- Use of octupling: simpler iteration also!
- η_T pairing: Miller function is $f_{\pm 2^{(3m+1)/2}-1, D_1}$

Constructing the Optimal Eta pairing

Algorithm	Tate double & add	Tate octuple & add	Barreto *et al.* η_T pairing	Optimal Ate pairing
#iterations	$2m$	$\frac{2m}{3}$	$\frac{m}{2}$	$\frac{m}{6}$

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \operatorname{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings
- Use of octupling: simpler iteration also!
- η_T pairing: Miller function is $f_{\pm 2^{(3m+1)/2} - 1, D_1}$
- Optimal Ate pairing
  - distortion map ψ is much more complex
  - iterations would be roughly twice as expensive
  - optimal Ate pairing not considered here

Constructing the Optimal Eta pairing

Algorithm	Tate double & add	Tate octuple & add	Barreto *et al.* η_T pairing	This paper Optimal Eta pairing
#iterations	$2m$	$\frac{2m}{3}$	$\frac{m}{2}$	$\frac{m}{6}$

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \operatorname{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings
- Use of octupling: simpler iteration also!
- η_T pairing: Miller function is $f_{\pm 2^{(3m+1)/2}-1, D_1}$
- Optimal Ate pairing
    - distortion map ψ is much more complex
    - iterations would be roughly twice as expensive
    - optimal Ate pairing not considered here
- Optimal Eta pairing

Constructing the Optimal Eta pairing

Algorithm	Tate double & add	Tate octuple & add	Barreto *et al.* η_T pairing	This paper Optimal Eta pairing
#iterations	$2m$	$\frac{2m}{3}$	$\frac{m}{2}$	$\frac{m}{6}$

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \operatorname{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings
- Use of octupling: simpler iteration also!
- η_T pairing: Miller function is $f_{\pm 2^{(3m+1)/2}-1, D_1}$
- Optimal Ate pairing
  - distortion map ψ is much more complex
  - iterations would be roughly twice as expensive
  - optimal Ate pairing not considered here
- Optimal Eta pairing
  - cannot use 2^m-th power Verschiebung: does not act on the curve

Constructing the Optimal Eta pairing

Algorithm	Tate double & add	Tate octuple & add	Barreto *et al.* η_T pairing	This paper Optimal Eta pairing
#iterations	$2m$	$\frac{2m}{3}$	$\frac{m}{2}$	$\frac{m}{3}$

- Vanilla Tate pairing: $\log_2 \ell \approx \log_2 \# \mathrm{Jac}_C(\mathbb{F}_{2^m}) \approx 2m$ doublings
- Use of octupling: simpler iteration also!
- η_T pairing: Miller function is $f_{\pm 2^{(3m+1)/2}-1, D_1}$
- Optimal Ate pairing
  - distortion map ψ is much more complex
  - iterations would be roughly twice as expensive
  - optimal Ate pairing not considered here
- Optimal Eta pairing
  - cannot use 2^m-th power Verschiebung: does not act on the curve
  - but can use 2^{3m}-th power Verschiebung
  - 33% improvement compared to Barreto *et al.*'s work

Considering degenerate divisors

- Some protocols allow to choose the form of one or two input divisors
- Consider degenerate divisors of the form

$$(P) - (\mathcal{O})$$

  - only 2 coordinates in $\mathbb{F}_{2^m}$ to represent such a divisor (instead of 4 coordinates for a general one)
  - since octupling acts on the curve:

$$[8]((P) - (\mathcal{O})) = ([8]P) - (\mathcal{O})$$

  - we can work with a point!

Considering degenerate divisors

- Some protocols allow to choose the form of one or two input divisors
- Consider degenerate divisors of the form

$$(P) - (\mathcal{O})$$

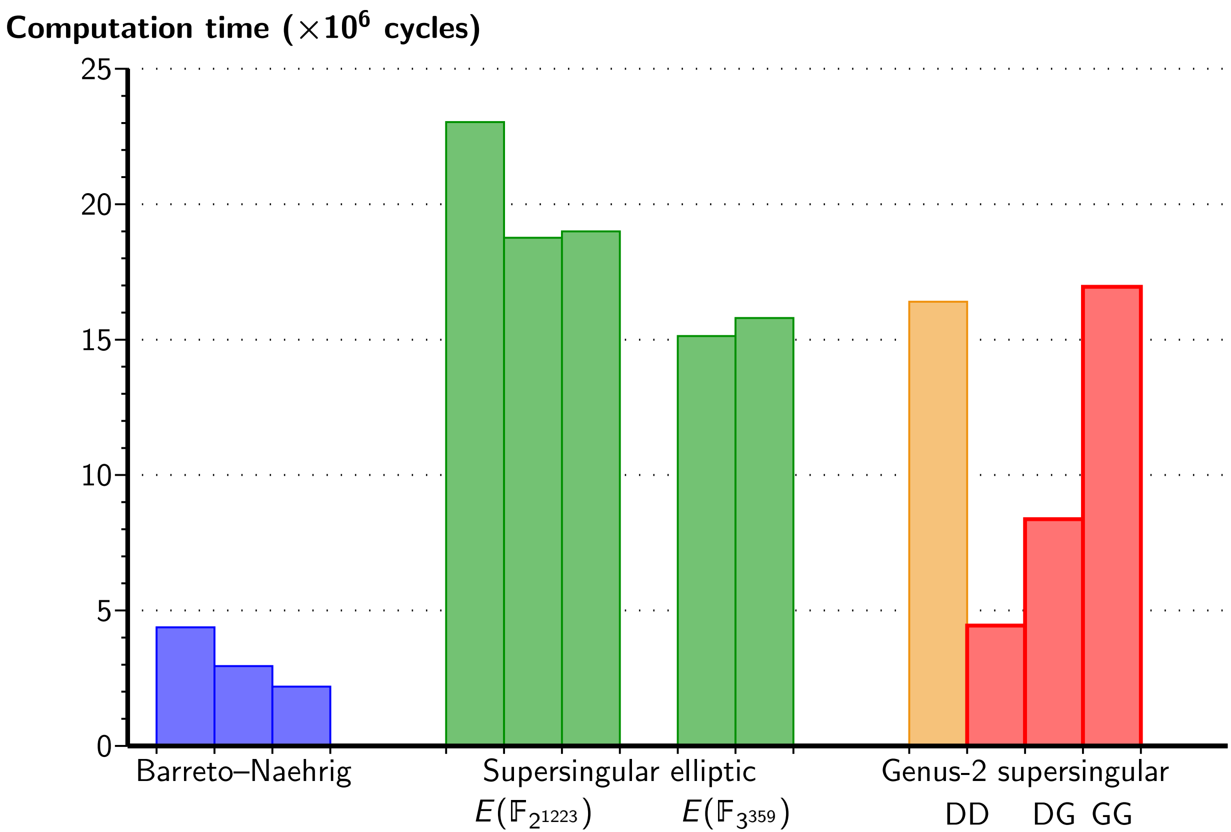
  - only 2 coordinates in $\mathbb{F}_{2^m}$ to represent such a divisor (instead of 4 coordinates for a general one)
  - since octupling acts on the curve:

$$[8]((P) - (\mathcal{O})) = ([8]P) - (\mathcal{O})$$

  - we can work with a point!
- We may compute the pairing of
  - two general divisors (GG)
  - one degenerate and one general divisor (DG)
    - ⋆ halves the amount of computation
    - ⋆ lot of protocols allow this
  - two degenerate divisors (DD)
    - ⋆ halves again the amount of computation
    - ⋆ some protocols still compatible

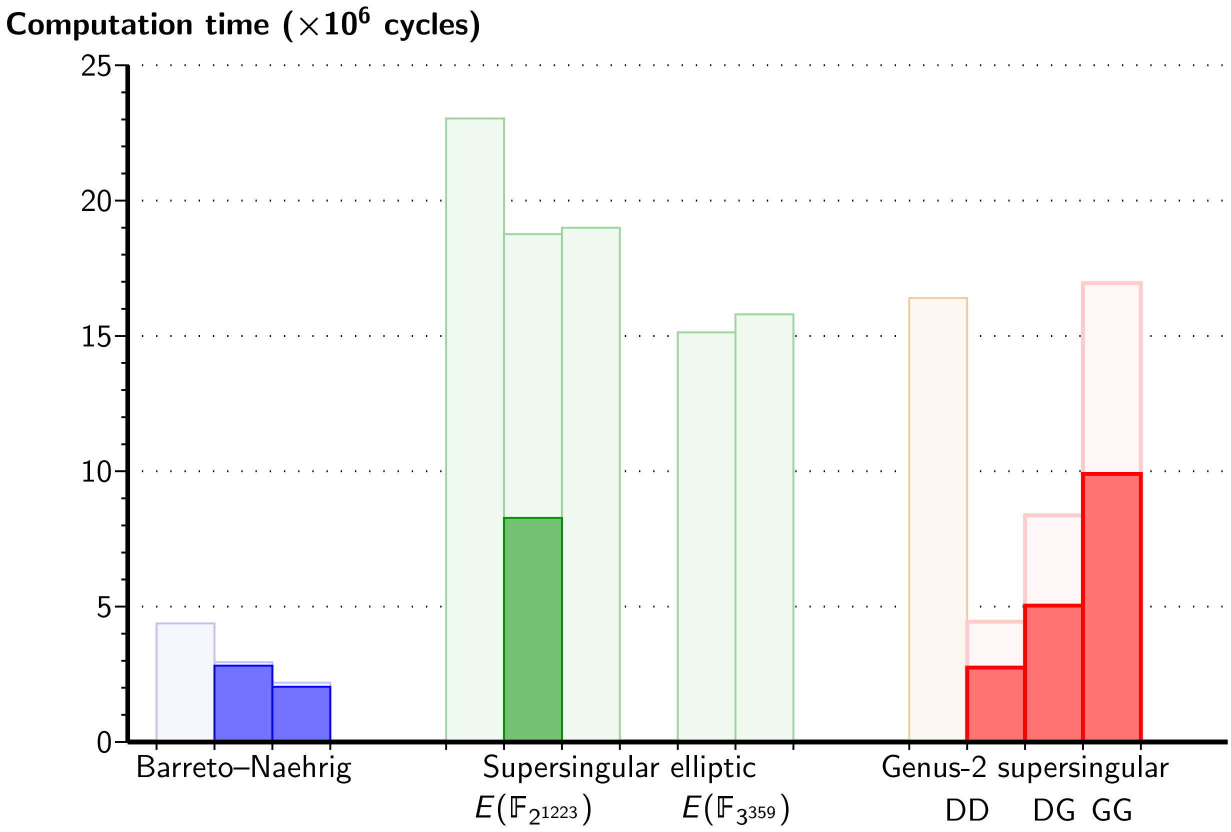
Software implementation

- Implementations for Intel Core 2



Software implementation

- Implementations for Intel Core 2 and Nehalem architecture
- Use of the native binary field multiplier on Nehalem



Hardware implementation

- Optimal Eta pairing on general divisors
- Implemented on a finite field coprocessor $\mathbb{F}_{2^{367}}$
  - addition
  - multiplication
  - Frobenius endomorphism
- Post place-and-route estimations on a Virtex 6-LX 130T results

Implementation	**Curve**	**Area (device usage)**	**Time (ms)**	**Area×time**
Cheung *et al.*	$E(\mathbb{F}_{p_{254}})$	35 %	0.57	4.03
Ghosh *et al.*	$E(\mathbb{F}_{2^{1223}})$	76 %	0.19	2.88
Estibals	$E(\mathbb{F}_{3^{5\cdot 97}})$	8 %	1.73	2.68
This work	$C_0(\mathbb{F}_{2^{367}})$ (GG)	7 %	3.09	4.30

Conclusion

- A novel pairing algorithm shortening Miller's loop
- Competitive timings compared to genus-1 pairings
- Comparable timings against non-symmetric pairings

Conclusion

- A novel pairing algorithm shortening Miller's loop
- Competitive timings compared to genus-1 pairings
- Comparable timings against non-symmetric pairings
- Most efficient symmetric pairing implementation
  - for both software and hardware
  - when at least one divisor is degenerate (DG and DD case)
- First hardware implementation of a genus-2 pairing reaching 128 bits of security

Conclusion

- A novel pairing algorithm shortening Miller's loop
- Competitive timings compared to genus-1 pairings
- Comparable timings against non-symmetric pairings
- Most efficient symmetric pairing implementation
  - for both software and hardware
  - when at least one divisor is degenerate (DG and DD case)
- First hardware implementation of a genus-2 pairing reaching 128 bits of security
- Perspectives
  - Implement optimal Ate pairing on this curve (work in progress)
  - Use theta functions for faster curve arithmetic

Thank you for your attention!

Questions?