Optimization strategies for hardware-based cofactorization

Benjamin Justus **Daniel Loebenberger**
Jens Putzka Joachim von zur Gathen

b-it, partially funded by the BSI

Factoring 2009, 11 September 2009

Goals

1. Optimize the COPACOBANA without changing the design.
2. After you have suceeded, analyze upcoming primitives like Edwards curves.

Contents

General optimizations

- Description of the model
- A toy example
- Further considerations

Optimizing the coordinate choices

- Edwards form
- Coordinate representations
- Use for the COPACOBANA

Conclusion

Contents

General optimizations
- Description of the model
- A toy example
- Further considerations

Optimizing the coordinate choices
- Edwards form
- Coordinate representations
- Use for the COPACOBANA

Conclusion

Known factoring algorithms

- Best known factorization algorithm for large numbers (above 300 bits, say) is the General Number Field Sieve (GNFS).
- In the so called Cofactorization Step a large number of smaller integers have to be factored.
- For these factorizations LENSTRA's Elliptic Curve Method (ECM) is used.

Fact:

Many runs of the same algorithm can be efficiently realized on a dedicated hardware cluster.

Known factoring algorithms

- Best known factorization algorithm for large numbers (above 300 bits, say) is the General Number Field Sieve (GNFS).
- In the so called Cofactorization Step a large number of smaller integers have to be factored.
- For these factorizations LENSTRA's Elliptic Curve Method (ECM) is used.

Fact:

Many runs of the same algorithm can be efficiently realized on a dedicated hardware cluster.

Components of a hardware cluster

Boards FPGAs ECM modules

Components of a hardware cluster

Boards FPGAs ECM modules

Components of a hardware cluster

Boards FPGAs ECM modules

Components of a hardware cluster

Boards FPGAs **ECM modules**

Components of a hardware cluster

Boards FPGAs ECM modules

Small modules $\Longrightarrow$ more parallelism, less inputs

Components of a hardware cluster

Boards FPGAs ECM modules

Large modules $\Longrightarrow$ less parallelism, more inputs

Components of a hardware cluster

Boards FPGAs ECM modules

Mixed modules not allowed!

Optimize!

Design goals

We want:

- A generic model of the cluster.
- A fast algorithm that computes the optimal distribution.
- A way of measuring the runtime achievement...
- ... against what?!?

Specifically in the case of the GNFS we need:

- Estimates on the number of parallel units per FPGA.
- Estimates on the average cost of one run of the ECM.
- A mathematical model of the ouputs of the GNFS: Difficult!

Design goals

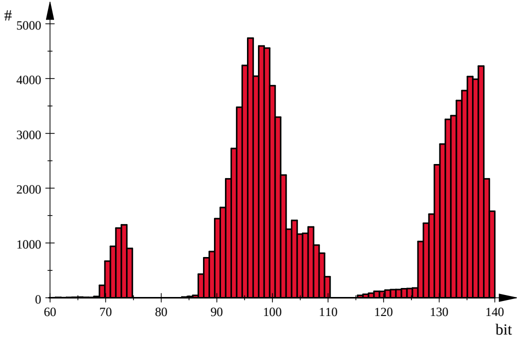
We want:

- A generic model of the cluster.
- A fast algorithm that computes the optimal distribution.
- A way of measuring the runtime achievement...
- ... against what?!?

Specifically in the case of the GNFS we need:

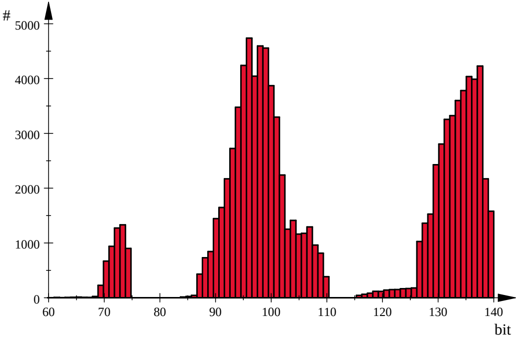
- Estimates on the number of parallel units per FPGA.
- Estimates on the average cost of one run of the ECM.
- A mathematical model of the ouputs of the GNFS: Difficult!

The GNFS produces numbers of different sizes:



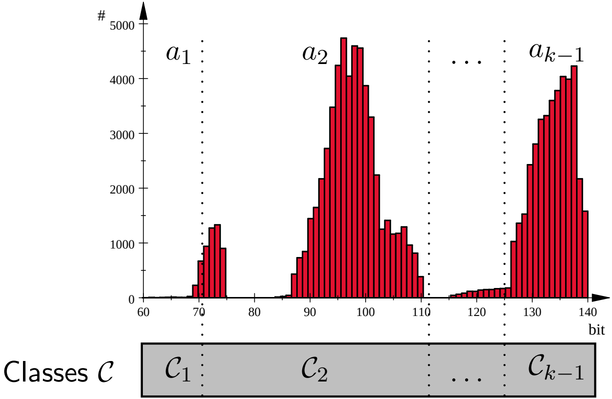
Classes

The GNFS produces numbers of different sizes:



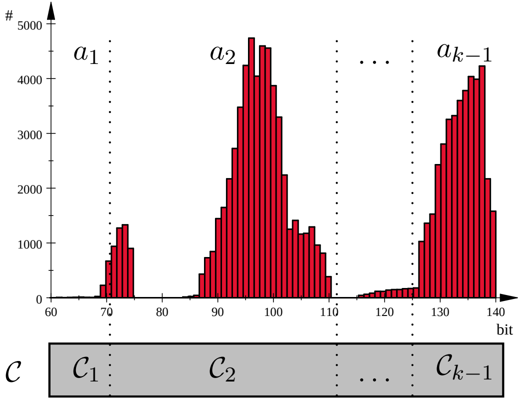
Classes $\mathcal{C}$

The GNFS produces numbers of different sizes:



Classes

The GNFS produces numbers of different sizes:

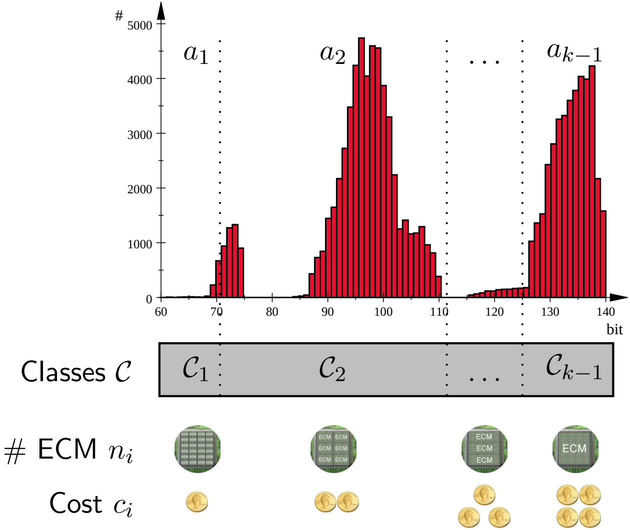


Classes $\mathcal{C}$

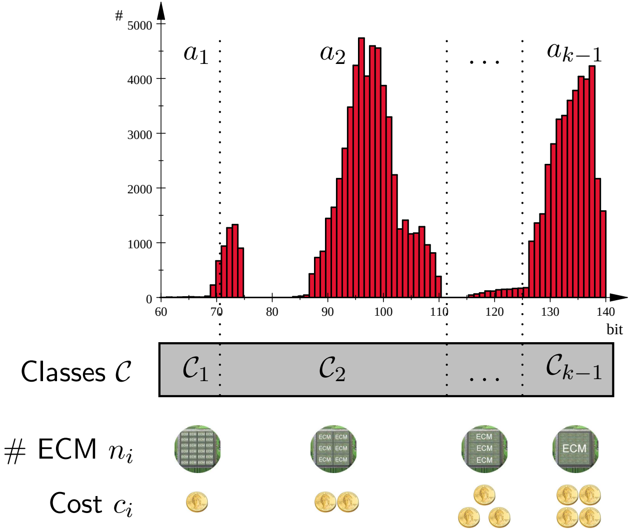
\# ECM n_i

Classes

The GNFS produces numbers of different sizes:



The GNFS produces numbers of different sizes:



Optimize over all possible classes!

Distribution on FPGAs

Given classes we need to fill many FPGAs:

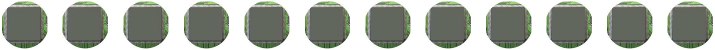
Classes $\mathcal{C}$ $\quad \mathcal{C}_1 \;\vdots\; \mathcal{C}_2 \;\vdots\; \ldots \;\vdots\; \mathcal{C}_{k-1}$

Distribution on FPGAs

Given classes we need to fill many FPGAs:

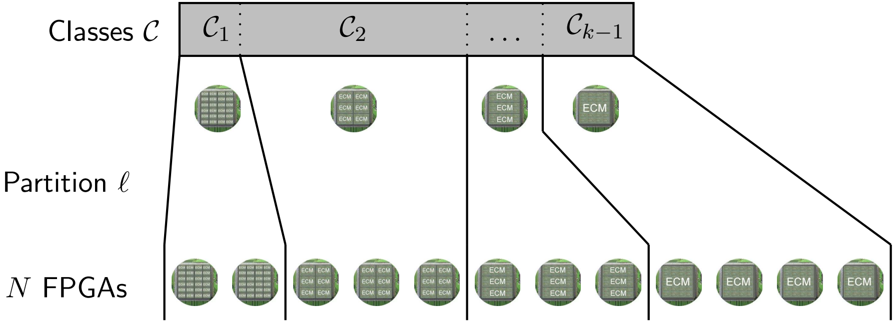
Classes $\mathcal{C}$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\dots$	$\mathcal{C}_{k-1}$

N FPGAs



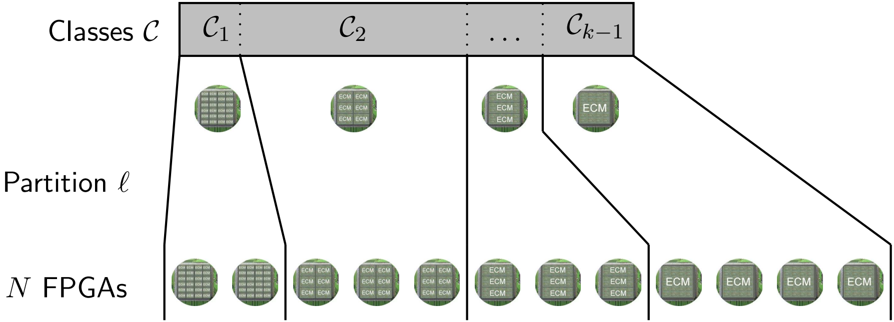
Distribution on FPGAs

Given classes we need to fill many FPGAs:



Distribution on FPGAs

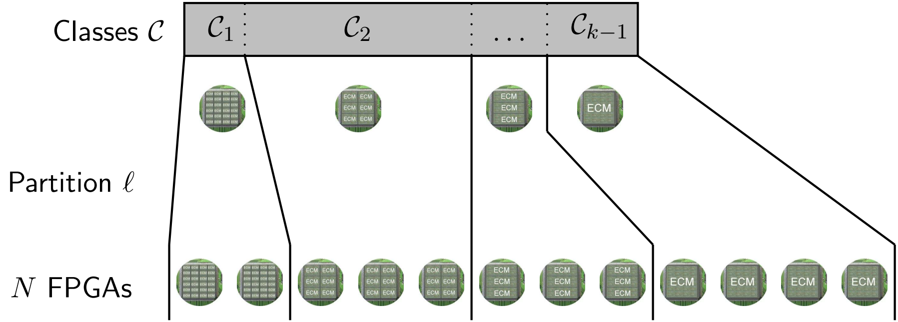
Given classes we need to fill many FPGAs:



Total cost of this configuration: $\max_{1 \leq i < k} \frac{c_i \cdot a_i}{n_i \cdot \ell_i}$.

Distribution on FPGAs

Given classes we need to fill many FPGAs:



Total cost of this configuration: $\max_{1 \leq i < k} \frac{c_i \cdot a_i}{n_i \cdot \ell_i}$.

Optimize over all configurations and classes, using e.g. BELLMAN's dynamic programming and a greedy heuristic!

A toy example

Assume we are constructing modules for $17i$-bit integers, given six data sets $\mathcal{D}_1, \ldots, \mathcal{D}_6$.

Number of parallel processes per chip:

Bitlength $17i$	51	68	85	102	119	136	153
Processes n_i	22	18	15	12	10	9	8

Average runtime of an ECM on the FPGA (in μs):

Bitlength $17i$	51	68	85	102	119
Cost c_i	856.35	1038.78	1221.22	1403.65	...

Optimize for 128 FPGAs.

A toy example

Assume we are constructing modules for $17i$-bit integers, given six data sets $\mathcal{D}_1, \ldots, \mathcal{D}_6$.

Number of parallel processes per chip:

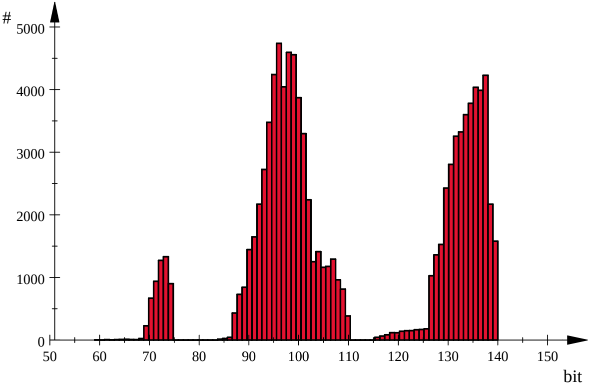
Bitlength $17i$	51	68	85	102	119	136	153
Processes n_i	22	18	15	12	10	9	8

Average runtime of an ECM on the FPGA (in μs):

Bitlength $17i$	51	68	85	102	119
Cost c_i	856.35	1038.78	1221.22	1403.65	...

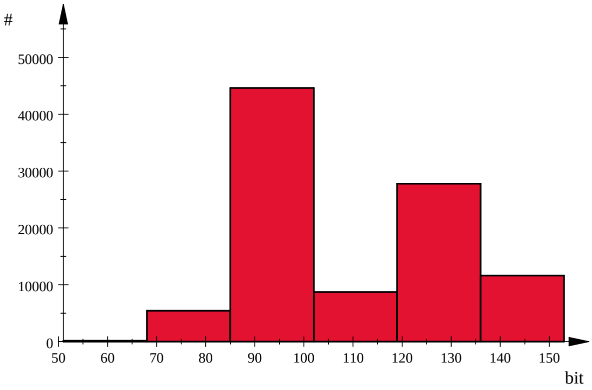
Optimize for 128 FPGAs.

Input distribution for dataset $\mathcal{D}_1$:



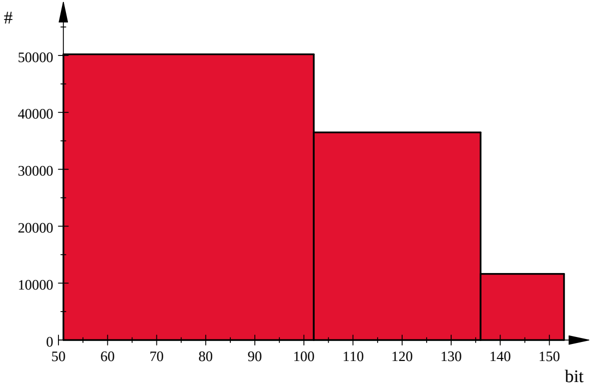
Results of the optimization

Distribution on the classes:



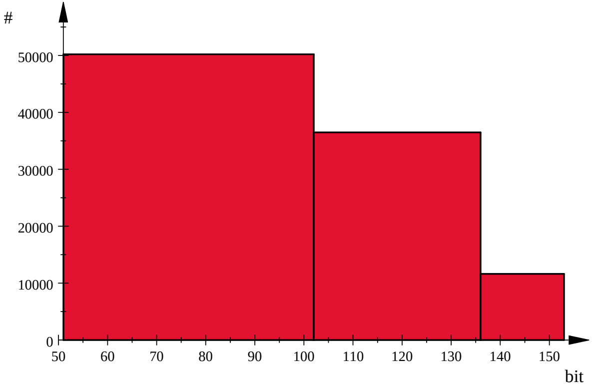
Results of the optimization

Optimal classes:



Results of the optimization

Optimal classes:



FPGAs: 47 58 23

Runtime gain

- Let $K \cdot g$ be the maximal size of the inputs.
- We compare the runtime to an unoptimized cluster.
- On such a cluster there are only modules for $K \cdot g$ bit numbers.
- The runtime is between

$$\sigma_{\mathcal{D}}^{-}(N,K) := \frac{1}{N}\sum_{i=1}^{K}\frac{c_i a_i}{n_K} \text{ and } \sigma_{\mathcal{D}}^{+}(N,K) := \frac{\#\mathcal{D}c_K}{N n_K}.$$

- Let $\gamma_{\mathcal{D}}^{\pm}$ denote the runtime gain of $\tau_{\mathcal{D}}$ against $\sigma_{\mathcal{D}}^{\pm}$ in percent.
- We obtain:

	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
$\gamma_{\mathcal{D}}^{-}$	17.47	16.97	17.66	18.38	18.4	16.88
$\gamma_{\mathcal{D}}^{+}$	33.29	32.73	33.36	33.86	33.5	32.12

Runtime gain

- Let $K \cdot g$ be the maximal size of the inputs.
- We compare the runtime to an unoptimized cluster.
- On such a cluster there are only modules for $K \cdot g$ bit numbers.
- The runtime is between

$$\sigma_{\mathcal{D}}^{-}(N,K) := \frac{1}{N}\sum_{i=1}^{K}\frac{c_i a_i}{n_K} \text{ and } \sigma_{\mathcal{D}}^{+}(N,K) := \frac{\#\mathcal{D} c_K}{N n_K}.$$

- Let $\gamma_{\mathcal{D}}^{\pm}$ denote the runtime gain of $\tau_{\mathcal{D}}$ against $\sigma_{\mathcal{D}}^{\pm}$ in percent.
- We obtain:

	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
$\gamma_{\mathcal{D}}^{-}$	17.47	16.97	17.66	18.38	18.4	16.88
$\gamma_{\mathcal{D}}^{+}$	33.29	32.73	33.36	33.86	33.5	32.12

Runtime gain

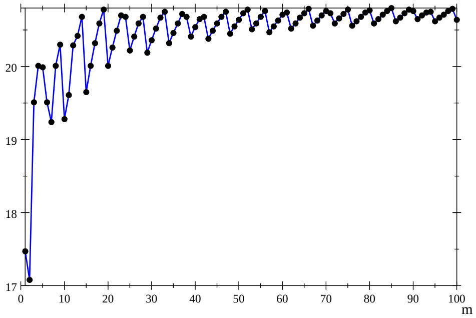
- Let $K \cdot g$ be the maximal size of the inputs.
- We compare the runtime to an unoptimized cluster.
- On such a cluster there are only modules for $K \cdot g$ bit numbers.
- The runtime is between

$$\sigma_{\mathcal{D}}^{-}(N,K) := \frac{1}{N}\sum_{i=1}^{K}\frac{c_i a_i}{n_K} \text{ and } \sigma_{\mathcal{D}}^{+}(N,K) := \frac{\#\mathcal{D} c_K}{N n_K}.$$

- Let $\gamma_{\mathcal{D}}^{\pm}$ denote the runtime gain of $\tau_{\mathcal{D}}$ against $\sigma_{\mathcal{D}}^{\pm}$ in percent.
- We obtain:

	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
$\gamma_{\mathcal{D}}^{-}$	17.47	16.97	17.66	18.38	18.4	16.88
$\gamma_{\mathcal{D}}^{+}$	33.29	32.73	33.36	33.86	33.5	32.12

We can use our model to optimize m clusters in parallel, obtaining:



	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_5$	$\mathcal{D}_6$
$\lim_{N\to\infty} \gamma_{\mathcal{D}}^{-}$	20.81	20.58	20.70	20.56	20.00	19.81
$\lim_{N\to\infty} \gamma_{\mathcal{D}}^{+}$	35.99	35.66	35.82	35.63	34.80	34.51

Online reconfiguration

- We have seen that the result of the optimization depends heavily on the input.
- On many clusters it is possible to reconfigure the FPGAs during the runtime of the cofactorization step.
- If we could use such a method, we could run a daemon on the controlling host computer that keeps track of the statistics of the inputs.
- The daemon has also to estimate when a reconfiguration makes sense.
- Further speedup possible!

To sum up

We have:

- A generic model of the cluster.
- A fast algorithm that computes the optimal distribution.
- A way of measuring the runtime achievement...
- ... against an unoptimized cluster.

Specifically in the case of the GNFS we have:

- Estimates on the number of parallel units per FPGA.
- Estimates on the average cost of one run of the ECM.
- But: No mathematical model of the ouputs of the GNFS!

Thus regardless of the coordinate choices we can optimize the cofactorization step in hardware considerably.

To sum up

We have:

- A generic model of the cluster.
- A fast algorithm that computes the optimal distribution.
- A way of measuring the runtime achievement...
- ... against an unoptimized cluster.

Specifically in the case of the GNFS we have:

- Estimates on the number of parallel units per FPGA.
- Estimates on the average cost of one run of the ECM.
- But: No mathematical model of the ouputs of the GNFS!

Thus regardless of the coordinate choices we can optimize the cofactorization step in hardware considerably.

To sum up

We have:

- A generic model of the cluster.
- A fast algorithm that computes the optimal distribution.
- A way of measuring the runtime achievement...
- ... against an unoptimized cluster.

Specifically in the case of the GNFS we have:

- Estimates on the number of parallel units per FPGA.
- Estimates on the average cost of one run of the ECM.
- But: No mathematical model of the ouputs of the GNFS!

Thus regardless of the coordinate choices we can optimize the cofactorization step in hardware considerably.

Contents

General optimizations
  Description of the model
  A toy example
  Further considerations

Optimizing the coordinate choices
  Edwards form
  Coordinate representations
  Use for the COPACOBANA

Conclusion

Motivation

Edwards introduced in 2007 a new normal form for elliptic curves.

First Question

Do we have a speedup for addition and multiplication?

Second Question

Can these curves be used on the COPACOBANA?

Edwards curves

- Let K be a field of characteristic $p \neq 2$.
- Then many elliptic curves C are birationally equivalent to a curve of the form

$$x^2 + y^2 = 1 + dx^2y^2$$

where $d \notin \{0, 1\}$.
- If C is not equivalent to a curve in Edwards form, then its quadratic twist is.
- In this case one needs to quadratically enlarge K.

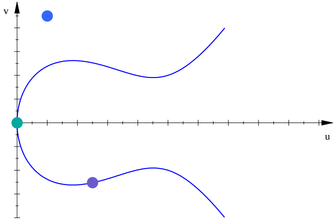
Edwards curves

- Let K be a field of characteristic $p \neq 2$.
- Then many elliptic curves C are birationally equivalent to a curve of the form

$$x^2 + y^2 = 1 + dx^2y^2$$

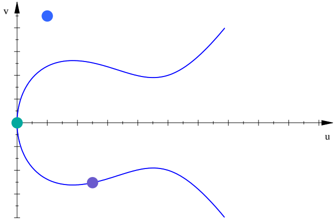
  where $d \notin \{0, 1\}$.
- If C is not equivalent to a curve in Edwards form, then its quadratic twist is.
- In this case one needs to quadratically enlarge K.

Transforming to Edwards form

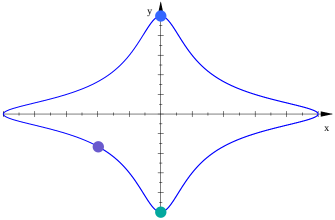


$$\frac{v^2}{1-d} = u^3 + 2\frac{1+d}{1-d}u^2 + u$$

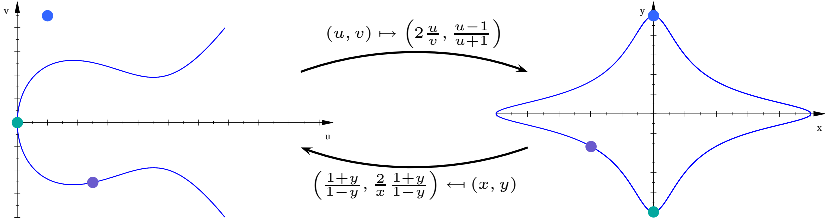
Transforming to Edwards form



$$\frac{v^2}{1-d} = u^3 + 2\frac{1+d}{1-d}u^2 + u$$



$$x^2 + y^2 = 1 + dx^2y^2$$



$\frac{v^2}{1-d} = u^3 + 2\frac{1+d}{1-d}u^2 + u$

$x^2 + y^2 = 1 + dx^2y^2$

Addition on Edwards curves

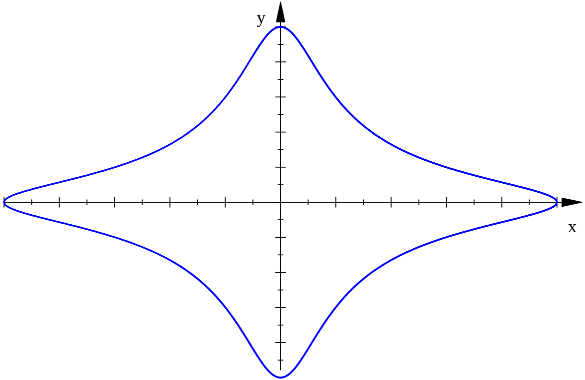
- We can define the following unified addition law:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + y_1 x_2}{1 + d x_1 x_2 y_1 y_2}, \frac{y_1 y_2 - x_1 x_2}{1 - d x_1 x_2 y_1 y_2} \right).$$

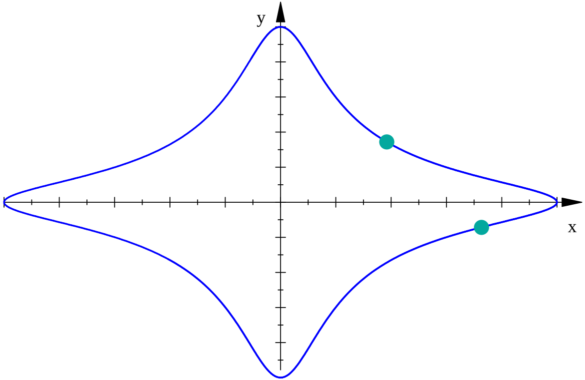
- Addition is well-defined if d is a nonsquare in the groundfield.
- The point $(0, 1)$ is the neutral element with respect to this addition law.
- Inverse of a point (x, y) is given by

$$-(x, y) = (-x, y).$$

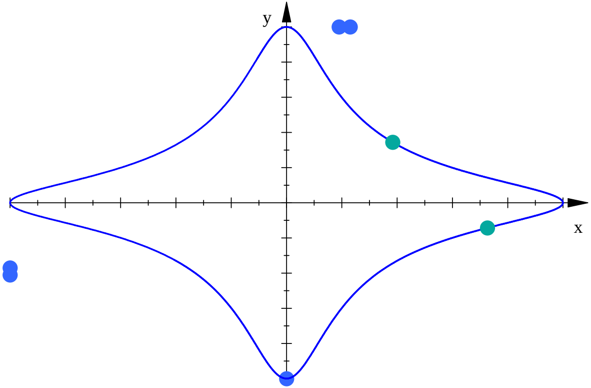
Addition visualized

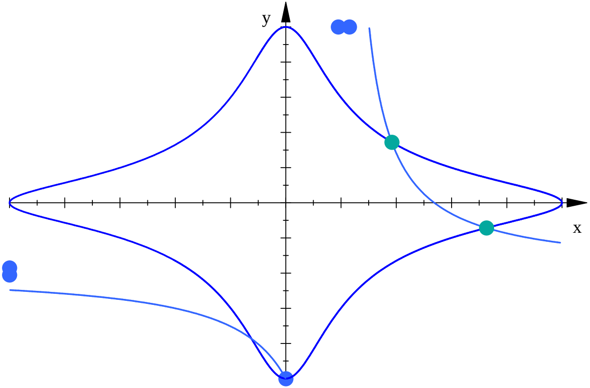


Addition visualized

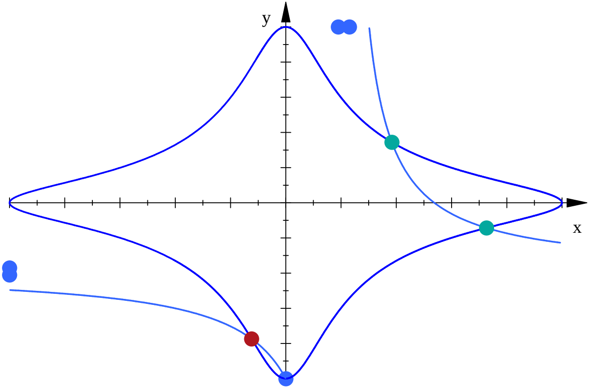


Addition visualized

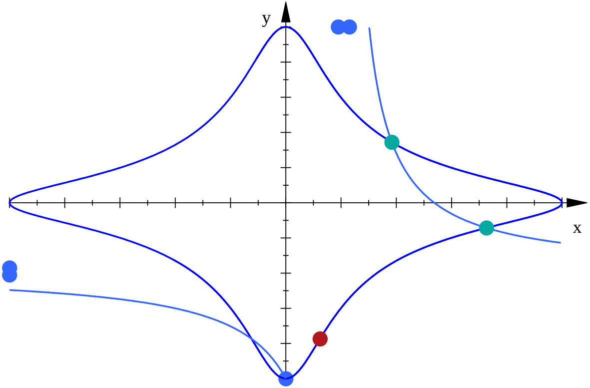



y
x

Addition visualized



Addition visualized



Twisted Edwards curves

- In 2008, Bernstein et al. introduced twisted Edwards curves.
- Prevents that we have to enlarge the ground field.
- A twisted Edwards curve $E_{a,d}$ of the curve E is given by the equation
$$ax^2 + y^2 = 1 + dx^2y^2.$$
- Here $a \neq d$ are both nonzero.
- Note that for parameters a_1, a_2, d_1, d_2 the curves E_{a_1,d_1} and E_{a_2,d_2} are quadratic twists of each other if $a_1d_2 = a_2d_1$.
- If further a_1/a_2 a square in K, then the curves are isomorphic, e.g. using the map
$$(x, y) \mapsto (\sqrt{a_1/a_2} \cdot x, y)$$

Twisted Edwards curves

- In 2008, Bernstein et al. introduced twisted Edwards curves.
- Prevents that we have to enlarge the ground field.
- A twisted Edwards curve $E_{a,d}$ of the curve E is given by the equation
$$ax^2 + y^2 = 1 + dx^2y^2.$$
- Here $a \neq d$ are both nonzero.
- Note that for parameters a_1, a_2, d_1, d_2 the curves E_{a_1,d_1} and E_{a_2,d_2} are quadratic twists of each other if $a_1 d_2 = a_2 d_1$.
- If further a_1/a_2 a square in K, then the curves are isomorphic, e.g. using the map
$$(x, y) \mapsto (\sqrt{a_1/a_2} \cdot x, y)$$

Available coordinate choices

- Bernstein and Lange introduced different types of Edwards coordinates.
- Clearly this choice is crucial for the speed of the basic operations on the curve.
- We need to analyze the different types of coordinates to each other.
- Also we have to compare these to well-known coordinates (for classical elliptic curves).
- As usual: Count cost in terms of basic operations.

Operation in the ground field	Cost
Multiplication	$\mathbf{M}$
Squaring	$\mathbf{S}$
Multiplication with a constant	$\mathbf{D}$
Addition	$\mathbf{A}$

Available coordinate choices

- Bernstein and Lange introduced different types of Edwards coordinates.
- Clearly this choice is crucial for the speed of the basic operations on the curve.
- We need to analyze the different types of coordinates to each other.
- Also we have to compare these to well-known coordinates (for classical elliptic curves).
- As usual: Count cost in terms of basic operations.

Operation in the ground field	Cost
Multiplication	$\mathbf{M}$
Squaring	$\mathbf{S}$
Multiplication with a constant	$\mathbf{D}$
Addition	$\mathbf{A}$

Projective Edwards coordinates

- Affine point on an Edwards curve E is given by a tuple (x_1, y_1).
- Projectively the curve has the homogenized form

$$X^2Z^2 + Y^2Z^2 = Z^4 + dX^2Y^2$$

- Here the projective point $(X : Y : Z)$ with $Z \neq 0$ corresponds to the affine point $(X/Z, Y/Z)$.

Cost

General addition: $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D} + 7\mathbf{A}$.

Doubling: $3\mathbf{M} + 4\mathbf{S} + 6\mathbf{A}$.

Projective Edwards coordinates

- Affine point on an Edwards curve E is given by a tuple (x_1, y_1).
- Projectively the curve has the homogenized form

$$X^2Z^2 + Y^2Z^2 = Z^4 + dX^2Y^2$$

- Here the projective point $(X : Y : Z)$ with $Z \neq 0$ corresponds to the affine point $(X/Z, Y/Z)$.

Cost

General addition: $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{D} + 7\mathbf{A}$.

Doubling: $3\mathbf{M} + 4\mathbf{S} + 6\mathbf{A}$.

Inverted Edwards coordinates

For Inverted Edwards Coordinates a projective point $(X : Y : Z)$ corresponds to the affine point $(Z/X, Y/X)$.

Cost

General addition: $9\mathbf{M} + 1\mathbf{S} + 1\mathbf{D} + 7\mathbf{A}$.

Doubling: $3\mathbf{M} + 4\mathbf{S} + 1\mathbf{D} + 6\mathbf{A}$.

Speedup of one multiplication for general addition, but one more multiplication with a constant for doubling!

Inverted Edwards coordinates

For Inverted Edwards Coordinates a projective point $(X : Y : Z)$ corresponds to the affine point $(Z/X, Y/X)$.

Cost

General addition: $9\mathbf{M} + 1\mathbf{S} + 1\mathbf{D} + 7\mathbf{A}$.

Doubling: $3\mathbf{M} + 4\mathbf{S} + 1\mathbf{D} + 6\mathbf{A}$.

Speedup of one multiplication for general addition, but one more multiplication with a constant for doubling!

Montgomery coordinates

- On a Mongomery curve $By^2 = x^3 + Ax^2 + x$ a point (x, y) is represented by a pair $(X\colon Z)$ such that $X/Z = x$.
- This representation does not distinguish (x, y) from $(x, -y)$!
- Thus $P + Q$ can only be computed if one knows

$$P, Q \text{ and } P - Q.$$

Cost

General addition: $6\mathbf{M} + 4\mathbf{S} + 1\mathbf{D}$.

Doubling: Same.

Montgomery coordinates

- On a Mongomery curve $By^2 = x^3 + Ax^2 + x$ a point (x, y) is represented by a pair $(X : Z)$ such that $X/Z = x$.
- This representation does not distinguish (x, y) from $(x, -y)$!
- Thus $P + Q$ can only be computed if one knows

$$P, Q \text{ and } P - Q.$$

Cost

General addition: $6\mathbf{M} + 4\mathbf{S} + 1\mathbf{D}$.

Doubling: Same.

Comparison to Edwards coordinates

- Montgomery curves have a faster differential point addition.
- Differential addition chains in general longer than standard ones.
- When using projective Edwards coordinates on twisted Edwards curves: Speedup possible.

Example

Scalar multiplication $[s]P$ with 256 bit scalar s:
Montgomery coordinates: On avg. $6\mathbf{M} + 4\mathbf{S} + 1\mathbf{D}$ per bit.
Edwards coordinates: On avg. $4.86\mathbf{M} + 4.12\mathbf{S} + 0.194\mathbf{D}$ per bit.
Speedup of $1.14\mathbf{M} - 0.12\mathbf{S} + 0.803\mathbf{D}$.

Comparison to Edwards coordinates

- Montgomery curves have a faster differential point addition.
- Differential addition chains in general longer than standard ones.
- When using projective Edwards coordinates on twisted Edwards curves: Speedup possible.

Example

Scalar multiplication $[s]P$ with 256 bit scalar s:
Montgomery coordinates: On avg. $6\mathbf{M} + 4\mathbf{S} + 1\mathbf{D}$ per bit.
Edwards coordinates: On avg. $4.86\mathbf{M} + 4.12\mathbf{S} + 0.194\mathbf{D}$ per bit.
Speedup of $1.14\mathbf{M} - 0.12\mathbf{S} + 0.803\mathbf{D}$.

Use of Edwards coordinates for the COPACOBANA

- During the GNFS we will factor most of the time smallish numbers.
- For such numbers the Edwards form does not yet yield any speedup.
- Additionally: High development cost if one wants to redesign the ECM modules!
- Thus it seems that for the ECM on the COPACOBANA the use of Edwards coordinates does not (yet) make sense.

Problem

So what can we do?

Use of Edwards coordinates for the COPACOBANA

- During the GNFS we will factor most of the time smallish numbers.
- For such numbers the Edwards form does not yet yield any speedup.
- Additionally: High development cost if one wants to redesign the ECM modules!
- Thus it seems that for the ECM on the COPACOBANA the use of Edwards coordinates does not (yet) make sense.

Problem

So what can we do?

Finding new constructions

Ideas:

- Exploiting the additional symmetry of Edwards curve, i.e. the operation $(x, y) \to (x, -y)$.
- Can we use existing constructions for Weierstraß curves in the case of Edwards curves?
- Can we improve the addition chains for ECM?

Finding new constructions

Ideas:

- Exploiting the additional symmetry of Edwards curve, i.e. the operation $(x, y) \to (x, -y)$.
- Can we use existing constructions for Weierstraß curves in the case of Edwards curves?
- Can we improve the addition chains for ECM?

Finding new constructions

Ideas:

- Exploiting the additional symmetry of Edwards curve, i.e. the operation $(x, y) \to (x, -y)$.
- Can we use existing constructions for Weierstraß curves in the case of Edwards curves?
- Can we improve the addition chains for ECM?

Conclusion

Optimizations for the cluster:

- Independent of the implementation as long as it is scalable.
- Independent of the concrete coordinate choices.
- Already here a speedup of up to 30%.

Optimizations of the ECM modules:

- Using optimized coordinates.
- Problem: High development cost!

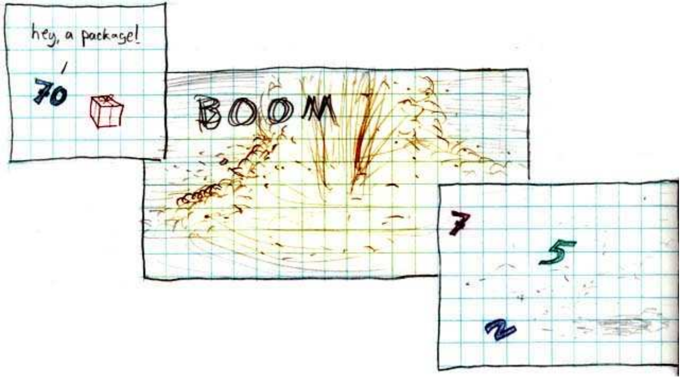
Conclusion

Optimizations for the cluster:

- Independent of the implementation as long as it is scalable.
- Independent of the concrete coordinate choices.
- Already here a speedup of up to 30%.

Optimizations of the ECM modules:

- Using optimized coordinates.
- Problem: High development cost!



Thank you!