Countermeasures against Differential Power Analysis for Hyperelliptic Curve Cryptosystems

Roberto Avanzi
mocenigo@exp-math.uni-essen.de

IEM – University of Duisburg–Essen

partially supported by the EU via the AREHCC Project
http://www.arehcc.com

Slide index

In full screen mode, click on titles to go to corresponding slide.

- Why HECC
- Group of Divisors
- Divisor Classes and Mumford Representation
- Dangers for HECC
  - Simple Side Channel Analysis
  - Differential Side Channel Analysis and countermeasures
  - Goubin-type attacks and countermeasures
- Typos in Paper
- Conclusions

Why HECC

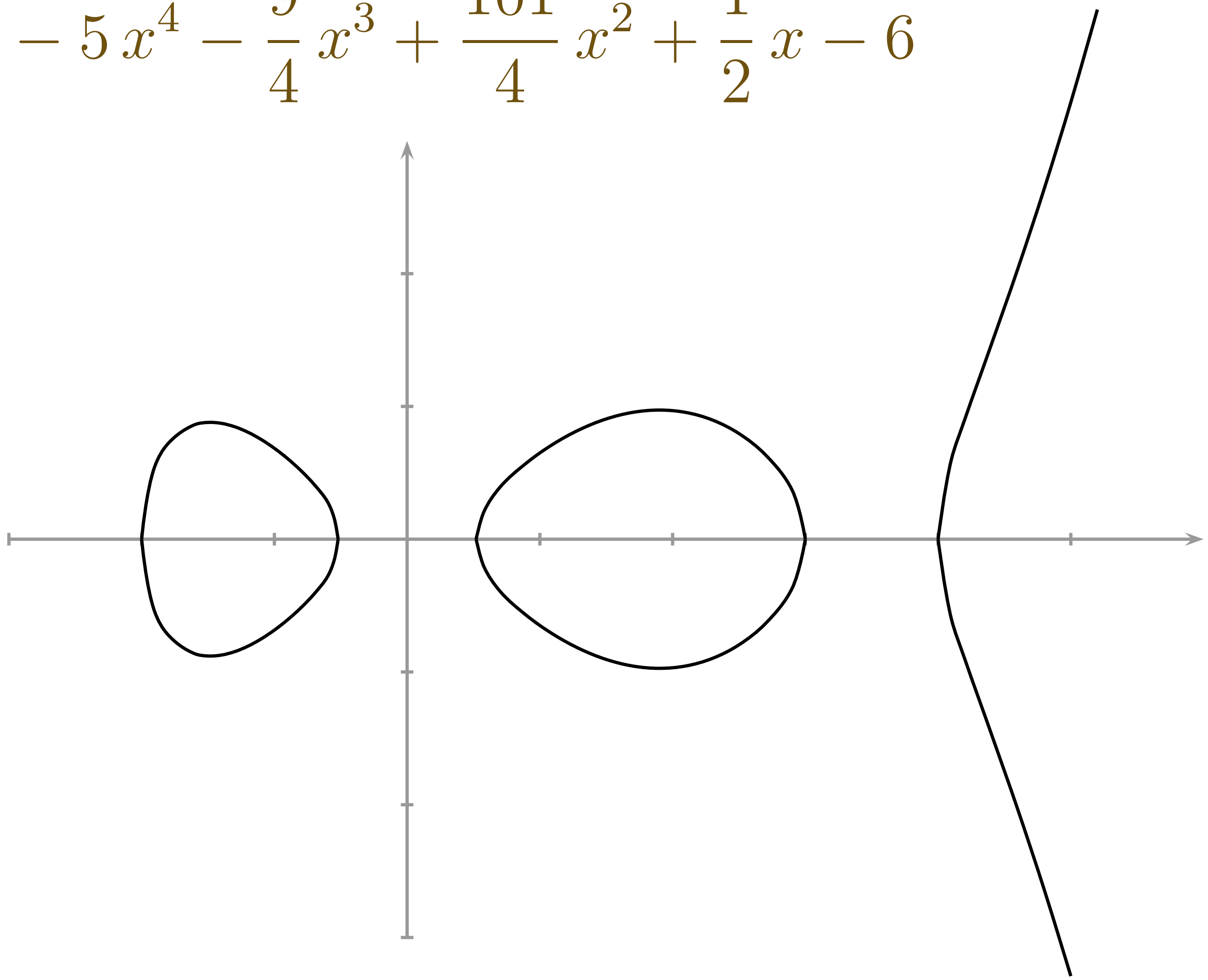
- Not wise to put all eggs in one basket.
- hecc close to ecc in performance: $\pm 10\%$.

  See Pelzl, et al., before lunch, for $g = 3$ over binary fields, and (in progress) A. for $g = 2$ over prime fields.
- Smaller fields might allow use of cheaper hardware (but: put more software on card.)
- For the moment, less patents for hecc than on ecc.

Just a reminder...

Here's how a hyperelliptic curve C of genus 2 looks like!

$$y^2 = x^5 - 5\,x^4 - \frac{9}{4}\,x^3 + \frac{101}{4}\,x^2 + \frac{1}{2}\,x - 6$$



Curve $\mathcal{C} \,:\, y^2 + h(x)y = f(x)$

f monic, $\deg f = 2g+1$, $\deg h \leqslant g$. g = genus.

Points on a hyperelliptic curve in general do not form a group!

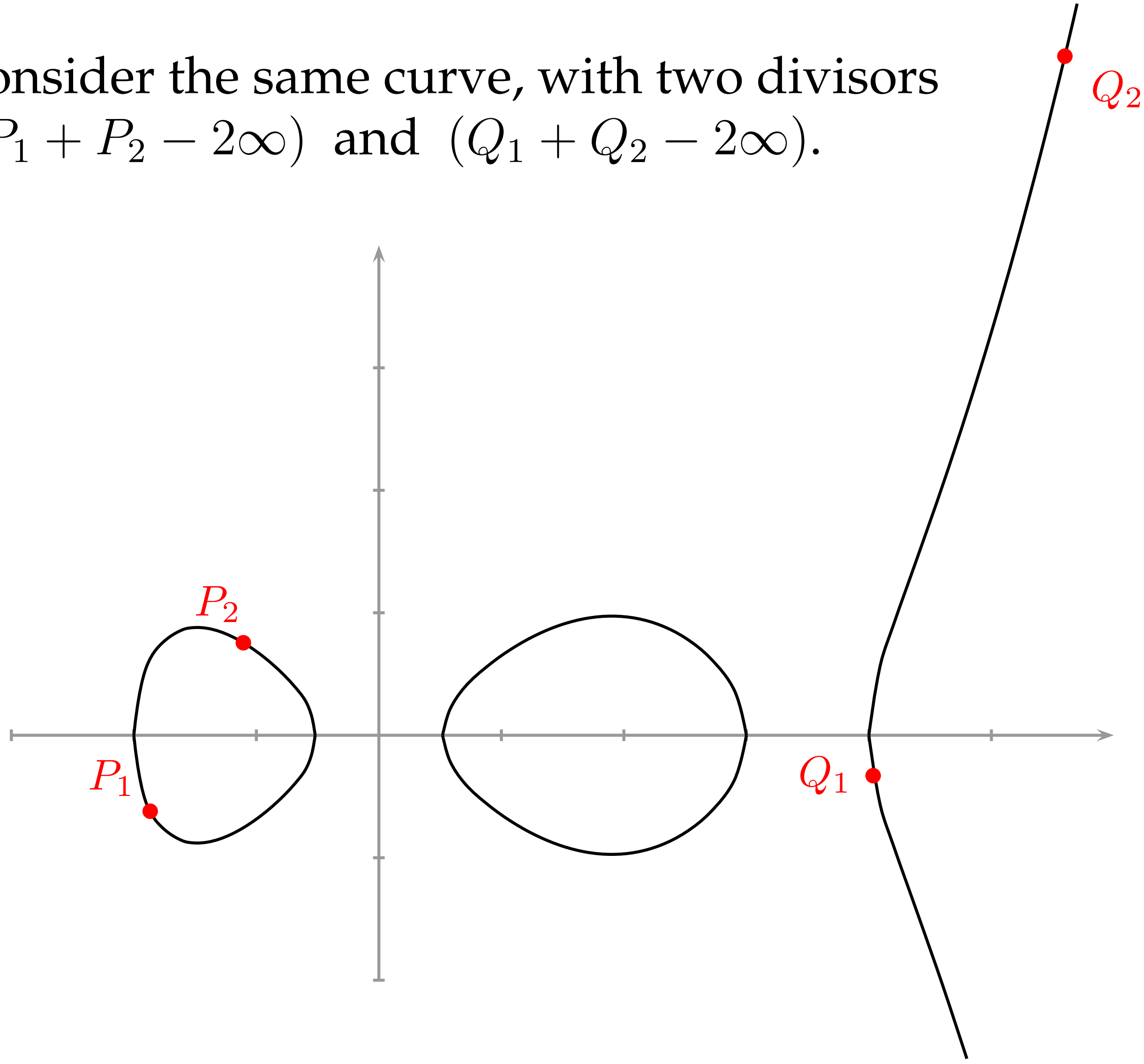
Use divisors, i. e. "sets of points" with multiplicities:

$$\sum_{i=1}^{k} m_i P_i - \left(\sum_{i=1}^{k} m_i \right) \infty \quad : \quad m_i > 0, \quad P_i \in \mathcal{C} \setminus \{\infty\}$$

We show how this works "geometrically".

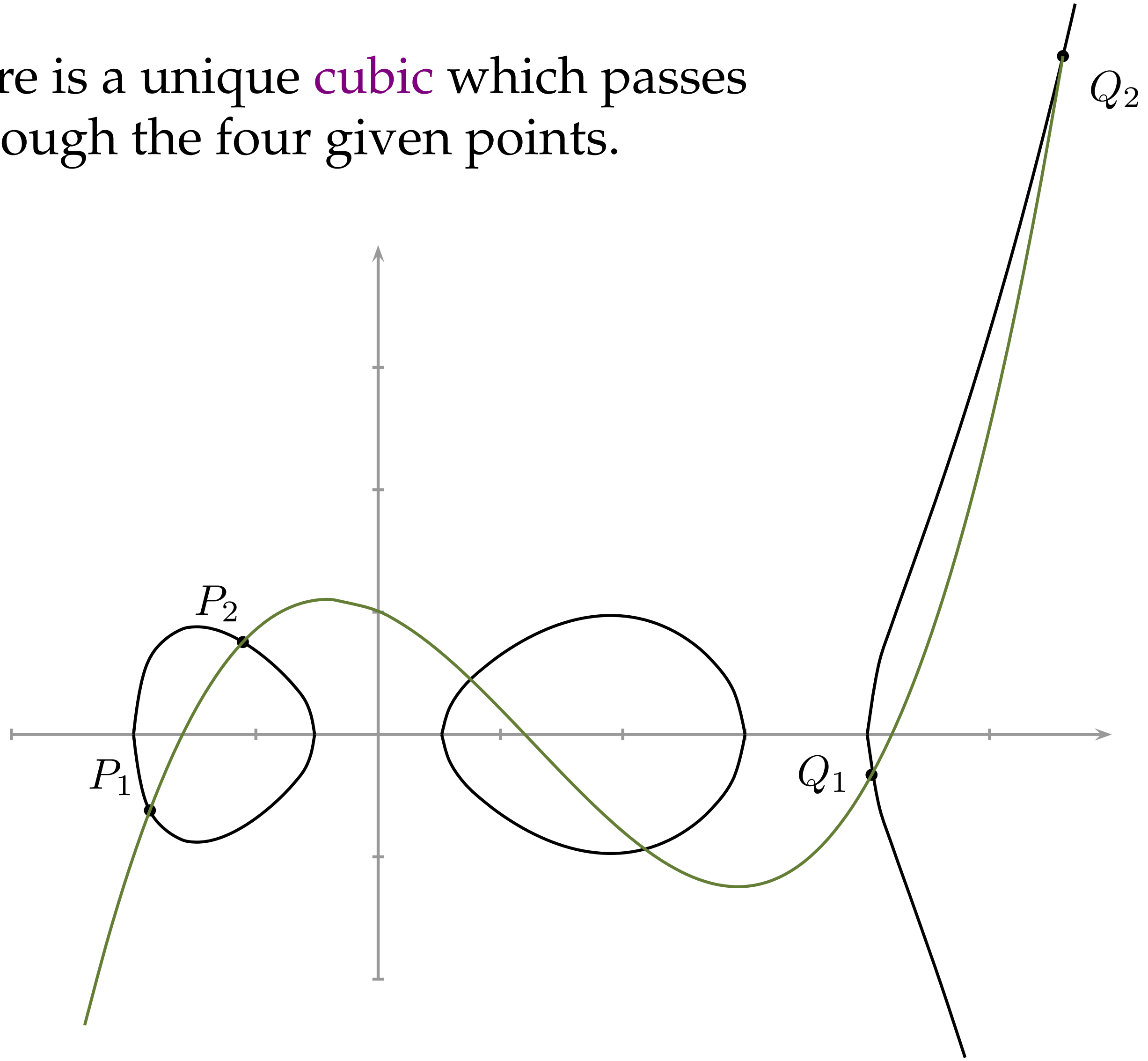
How to do HECC

Consider the same curve, with two divisors $(P_1 + P_2 - 2\infty)$ and $(Q_1 + Q_2 - 2\infty)$.



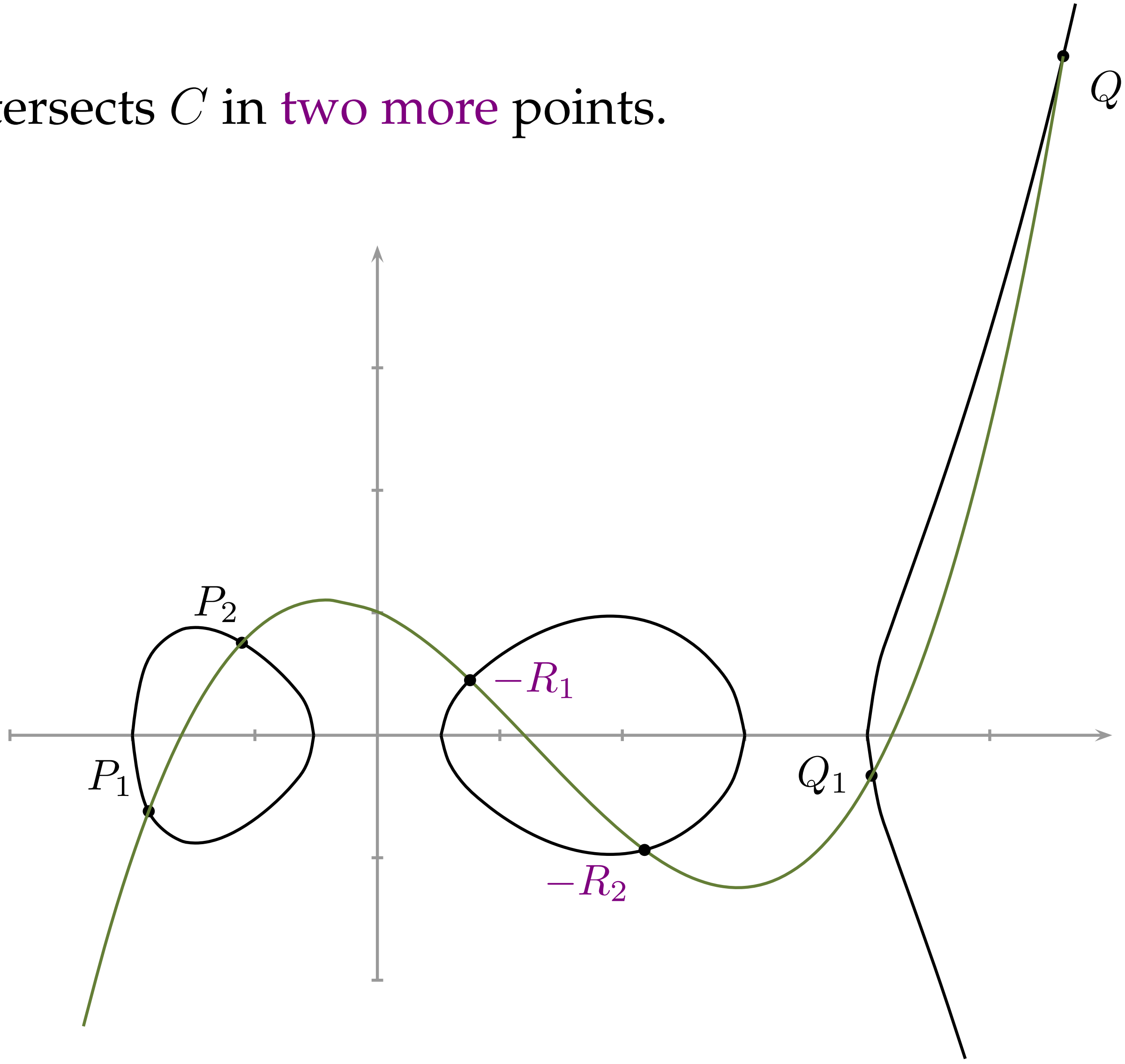
How to do HECC

There is a unique cubic which passes through the four given points.



How to do HECC

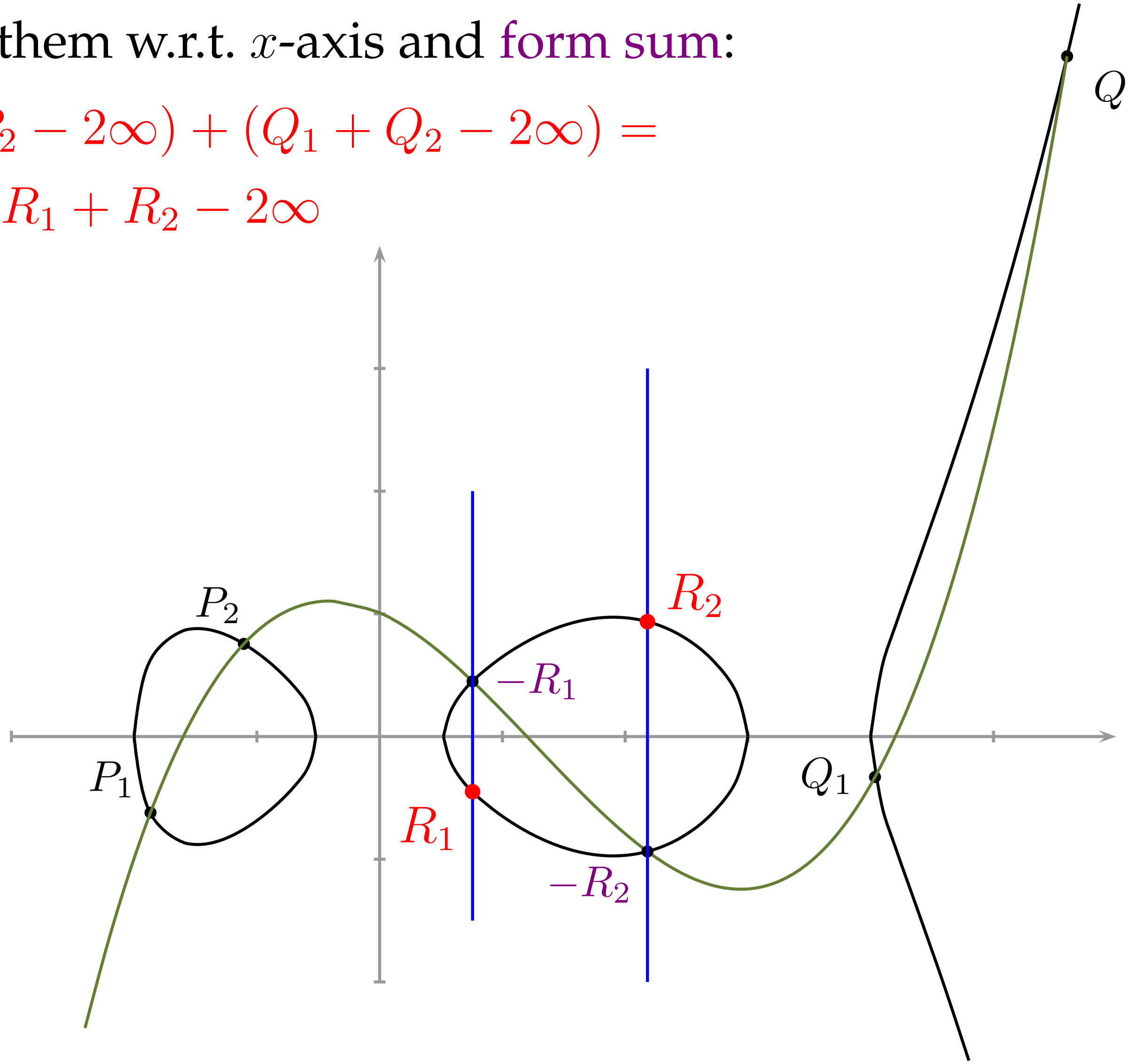
It intersects C in two more points.



How to do HECC

Mirror them w.r.t. x-axis and form sum:

$$(P_1 + P_2 - 2\infty) + (Q_1 + Q_2 - 2\infty) =$$
$$= R_1 + R_2 - 2\infty$$



Divisor Classes and Mumford Representation

This defines a group, the Jacobian of $\mathcal{C}$, $\mathrm{Jac}(\mathcal{C})$.

If $\mathcal{K} = \mathbb{F}_q$, then $\#\,\mathrm{Jac}(\mathcal{C}) \approx q^g$.

But working with “point sets” and intersecting curves is very inefficient.

Better:

- Mumford representation and
- Cantor’s algorithm $\Rightarrow$ explicit formulæ.

Divisor Classes and Mumford Representation

Curve $\mathcal{C} \; : \; y^2 + h(x)y = f(x)$

Let $D = \sum m_P P - \left(\sum m_P\right)\infty$ have deg $\sum m_P \leqslant g$.
– more precisely: degree of associated effective divisor –

D represented by unique pair of polynomials
$U(t), V(t) \in \mathcal{K}[t]$ with: $g \geqslant \deg_t U > \deg_t V$, U monic.

$$\begin{cases} U(t) = \prod (t - x_P)^{m_P} \\ V(x_P) = y_P \;\; \text{for all } P \\ U(t) \;\; \text{divides} \;\; V(t)^2 + V(t)h(t) - f(t) \end{cases}$$

Coordinates of D: the coefficients of U and V .

Two categories of attacks:

- *mathematical* (on structure) and
- *hardware-related* (on implementation).

Two categories of attacks:

- *mathematical* (on structure) and
- *hardware-related* (on implementation).

Mathematical attacks $\Rightarrow$ $g \leqslant 4$.

Two categories of attacks:

- *mathematical* (on structure) and
- *hardware-related* (on implementation).

Mathematical attacks $\Rightarrow$ $g \leqslant 4$.

Here we consider *Side Channel Analysis*.

- Simple.
- Differential.
- Goubin type.

I will not *describe them for the umpteenth time here...*

Not *interested in fault analysis in this paper.*

Solution: make sequence of elementary ops regular.

- Make sequence of group ops homogeneous (e.g. Coron's double-and-add-always).
- Make the group ops indistinguishable (e.g. Hess or Jacobi form for `ecc`, Brier-Joye, insertion of dummy ops: latter easy with Lange's genus 2 formulae); or split the group ops into blocks which can be made regular (Ciet-Joye).

From now assume hecc immunised against SPA.

Differential Side Channel Analysis

Applies to computations $n \cdot D$ in the group G, n fixed.

Exploits knowledge of internal representation of operands.

$\Rightarrow$ **internal data must be unpredictably scrambled:**

Some techniques for previous cryptosystems:

Faster

- Joye-Tymen (`ecc`): Compute in isomorphic curve.
- Coron's 2nd and 3rd (`ecc`): Randomise D.

Slower

- Coron's 1rst: Randomise scalar n.
- Joye-Tymen: Use isomorphic binary field.

Which countermeasures for `hecc`*?*

First Countermeasure: Curve Randomisation

`hecc` analogue of Joye-Tymen's `ecc` curve randomisation.

$\phi : \mathcal{C} \to \tilde{\mathcal{C}}$ = a $\mathcal{K}$-isomorphism of hyperelliptic curves.

$\Rightarrow$ $\mathcal{K}$-isomorphism $\phi : \mathrm{Jac}(\mathcal{C}) \to \mathrm{Jac}\left(\tilde{\mathcal{C}}\right)$.

Assume ϕ and ϕ^{-1} can be computed "quickly".

First Countermeasure: Curve Randomisation

`hecc` analogue of Joye-Tymen's `ecc` curve randomisation.

$\phi : \mathcal{C} \to \tilde{\mathcal{C}}$ = a $\mathcal{K}$-isomorphism of hyperelliptic curves.

$\Rightarrow$ $\mathcal{K}$-isomorphism $\phi : \mathrm{Jac}(\mathcal{C}) \to \mathrm{Jac}\left(\tilde{\mathcal{C}}\right)$.

Assume ϕ and ϕ^{-1} can be computed "quickly".

Instead of $Q = n \cdot D$ in $\mathrm{Jac}(\mathcal{C})(\mathcal{K})$, we compute

$$Q = \phi^{-1}\big(n \cdot \phi(D)\big)$$

First Countermeasure: Curve Randomisation

`hecc` analogue of Joye-Tymen's `ecc` curve randomisation.

$\phi : \mathcal{C} \rightarrow \tilde{\mathcal{C}}$ = a $\mathcal{K}$-isomorphism of hyperelliptic curves.

$\Rightarrow$ $\mathcal{K}$-isomorphism $\phi : \mathrm{Jac}(\mathcal{C}) \rightarrow \mathrm{Jac}\left(\tilde{\mathcal{C}}\right)$.

Assume ϕ and ϕ^{-1} can be computed "quickly".

Instead of $Q = n \cdot D$ in $\mathrm{Jac}(\mathcal{C})(\mathcal{K})$, we compute

$$Q = \phi^{-1}\big(n \cdot \phi(D)\big)$$

Transfer all points of D over $\mathcal{C}$ to $\tilde{\mathcal{C}}$ "simultaneously" by manipulating coordinates of divisor.

First Countermeasure: Curve Randomisation

$$\begin{tikzcd}[column sep=huge] D \in \mathrm{Jac}(\mathcal{C})(\mathcal{K}) \arrow[r, "\text{multiplication by } n"] \arrow[d, "\phi"'] & \mathrm{Jac}(\mathcal{C})(\mathcal{K}) \ni n \cdot D \\ \phi(D) \in \mathrm{Jac}\left(\tilde{\mathcal{C}}\right)(\mathcal{K}) \arrow[r, "\text{multiplication by } n"'] & \mathrm{Jac}\left(\tilde{\mathcal{C}}\right)(\mathcal{K}) \ni n \cdot \phi(D) \arrow[u, "\phi^{-1}"'] \end{tikzcd}$$

Details in the paper. Two types of isomorphisms:

Using only multiplications: *All coefficients of $\mathcal{C}$ and of D are multiplied by different powers of a randomly chosen $s \in \mathcal{K}$.*
Total # field muls LESS than in one group op!

Using also additions: everything can become slower.
(work with more general curves).

Second Countermeasure: Divisor Randomisation

On embedded hardware, field inversion is very slow.

This prompted the introduction of projective coordinates.
They do not require inversions.
A group element has many different representations.

For `ecc`: two triples (X, Y, Z) and (sX, sY, sZ) represent the same point if $s \in \mathcal{K}^\times$.

Coron uses them to randomise the base point:
replaces (X, Y, Z) with (sX, sY, sZ) for a random $s \in \mathcal{K}^\times$.

Second Countermeasure: Divisor Randomisation

For genus 2 `hecc`: Projective and New coordinates (Lange).

Projective: a divisor $D \equiv [U(t), V(t)]$ is represented as a quintuple $[U_1, U_0, V_1, V_0, Z] \in \mathcal{K}^5$ where

$$U(t) = t^2 + \frac{U_1}{Z} t + \frac{U_0}{Z} \quad \text{and} \quad V(t) = \frac{V_1}{Z} t + \frac{V_0}{Z} \ .$$

The randomisation consists in picking a random $s \in \mathcal{K}^\times$ and by performing the following replacement

$$[U_1, U_0, V_1, V_0, Z] \mapsto [sU_1, sU_0, sV_1, sV_0, sZ] \ .$$

For New coordinates the method is entirely similar.

Goubin type attacks: Context

Remark: randomisation of zero by multiplication by a random value, or by random isomorphism, is... zero!

Definition: context of Goubin-type attacks:

Let H be a small subset of the group G s.t.:

- *The elements of H possess properties which makes their processing detectable by side-channel analysis – for example, zeros in the internal representation – and*
- *are invariant under a given randomisation procedure R.*

H := set of special points/divisors.

Goubin type attacks: Description

Suppose most significant digits $n_r, n_{r-1}, \ldots, n_{j+1}$ of n known; we want to find n_j.

Goubin type attacks: Description

Suppose most significant digits $n_r, n_{r-1}, \ldots, n_{j+1}$ of n known; we want to find n_j.

Assume that a *chosen message attack* can be set up to obtain *an element of H* as a partial result in a specific step of the scalar multiplication – if n_j has been guessed correctly.

Goubin type attacks: Description

Suppose most significant digits $n_r, n_{r-1}, \ldots, n_{j+1}$ of n known;
we want to find n_j.

Assume that a *chosen message attack* can be set up to obtain *an element of H* as a partial result in a specific step of the scalar multiplication – if n_j has been guessed correctly.

This element may be $t \cdot D$ where D is the chosen message and $t =$ number represented by $(n_r, n_{r-1}, \ldots, n_{j+1}, n_j)$.
In this case the specific step of the scalar multiplication would be an addition or doubling involving $t \cdot D$.
For other value(s) of n_j elements of H should be avoided.

Goubin type attacks: Description

Suppose most significant digits $n_r, n_{r-1}, \ldots, n_{j+1}$ of n known; we want to find n_j.

Assume that a *chosen message attack* can be set up to obtain *an element of H* as a partial result in a specific step of the scalar multiplication – if n_j has been guessed correctly.

This element may be $t \cdot D$ where D is the chosen message and $t =$ number represented by $(n_r, n_{r-1}, \ldots, n_{j+1}, n_j)$.
In this case the specific step of the scalar multiplication would be an addition or doubling involving $t \cdot D$.
For other value(s) of n_j elements of H should be avoided.

Then, statistical correlation of side-channel traces may reveal if the guess was correct even if R is used.

Goubin type attacks: The bad news

Such sets H exist. Examples:

- $H =$ Points with a zero coordinate of an elliptic curve.
- $H =$ Divisors on a hyperelliptic curve with a zero coordinate (e.g. of deg $< g$).

Preserved by above randomisations.

Probability random point/divisor $\in H$ is $O(q^{-1})$, $q = \#\mathcal{K}$, so set is small.

On `ecc` it is easy to avoid such points (remember Nigel Smart's talk). But for `hecc`?

Goubin type attacks: The good news

Scalar randomization: ok (but: slow).

Message blinding: `hecc` analogue of Coron's 2nd method.

R = secret divisor, with $S = n \cdot R$ known.

Compute $n \cdot (D + R) - S$ in place of $n \cdot D$.

If R belongs to the group generated by D (normal case), equivalent to isogeny of random degree:
if $R = m \cdot D$ then $D + R = (m + 1) \cdot D$.

An isogeny is *not* an isomorphism with probability $1 - O(q^{-1}) \Rightarrow$ images of "special" divisors are not special.

Page 378, line 7.

ERRATA: ... $\deg(t(D+R)) = g$ also with probability $O(q^{-1})$ ...

CORRIGE: ... $\deg(t(D+R)) < g$ also with probability $O(q^{-1})$ ...

Conclusions

- Two methods to prevent basic DPA for `hecc`.
  - Curve randomisation (generic).
  - Divisor randomisation (specific).
  - Cheaper than a single group operation!
- Serious Goubin-type attacks on `hecc` discovered.
  - Suitable divisor randomisation to thwart them.
  - Costs as few group operations.

Ditto for trace-zero varieties (Frey, Naumann, Lange, Lange-A.).

Now one can really start deploying `hecc` *on embedded devices!*

Any questions?