The Cryptographic Marriage of (Georg) Frobenius and Point Halving

Officiant: Roberto Avanzi
(Witnesses: Mathieu Ciet and Francesco Sica)
mocenigo@exp-math.uni-essen.de

IEM – University of Duisburg–Essen

Research partially supported by the EU via the Projects AREHCC (http://www.arehcc.com) and NESSIE (http://www.cryptonessie.org), and by the DFG.

Dedication

Dedicated to Preda Mihăilescu on occasion of the birth of his daughter Seraina Maria Teresa Sophia (Mihăilescu). (6 hours old in the photo.)

Outline of Talk and Slide index

With Acrobat Reader in full screen mode, click on titles to go to slide corresponding to desided topic.

As it often happens, important issues arise when a woman (Alice) wants to talk a man (Bob).

Alice and Bob want to agree on a common key for establishing secure (encrypted) communication over an insecure channel.

Bare-bones Diffie-Hellman Protocol

Given: a distinguished element P of a group Γ.

Alice		**Bob**
1. secretly picks $a < \#\langle P \rangle$		1. secretly picks $b < \#\langle P \rangle$
2. computes $Q_1 = aP$		2. computes $Q_2 = bP$
3. publishes Q_1		3. publishes Q_2
4. computes		4. computes

$$aQ_2 \quad = \quad abP \quad = \quad bQ_1$$

Common Key: the group element $K = (ab)P \in \langle P \rangle \subseteq \Gamma$

Bare-bones Diffie-Hellman Protocol

Given: a distinguished element P of a group Γ.

Alice		**Bob**
1. secretly picks $a < \#\langle \boldsymbol{P} \rangle$		1. secretly picks $b < \#\langle \boldsymbol{P} \rangle$
2. computes $Q_1 = a\boldsymbol{P}$		2. computes $Q_2 = b\boldsymbol{P}$
3. publishes Q_1		3. publishes Q_2
4. computes		4. computes

$$aQ_2 \quad = \quad ab\boldsymbol{P} \quad = \quad bQ_1$$

Common Key: the group element $K = (ab)\boldsymbol{P} \in \langle \boldsymbol{P} \rangle \subseteq \Gamma$

Crucial Computation: $s\boldsymbol{Q}$ given $s \in \mathbb{Z}$ and $\boldsymbol{Q} \in \Gamma$.

Bare-bones Diffie-Hellman Protocol

Given: a distinguished element P of a group Γ

Alice	Bob
1. secretly picks $a < \#\langle P\rangle$	1. secretly picks $b < \#\langle P\rangle$
2. computes $Q_1 = aP$	2. computes $Q_2 = bP$
3. publishes Q_1	3. publishes Q_2
4. computes	4. computes

$$aQ_2 = abP = bQ_1$$

Common Key: the group element $K = (ab)P \in \langle P\rangle \subseteq \Gamma$

Crucial Computation: sQ given $s \in \mathbb{Z}$ and $Q \in \Gamma$.

Version of protocol presented here insecure for authenticated key-exchange.

It can be made secure by modifying it.

But: the basic operation remains the computation of scalar products, i.e.

sQ given $s \in \mathbb{Z}$ and $Q \in \Gamma$.

Bare-bones Diffie-Hellman Protocol

Version of protocol presented here insecure for authenticated key-exchange.

It can be made secure by modifying it.

But: the basic operation remains the computation of scalar products, i.e.

sQ given $s \in \mathbb{Z}$ and $Q \in \Gamma$.

We now see some groups Γ and related scalar multiplications techniques which conjugate speed and (AFAWK) security.

$$E : y^2 + (a_1 x + a_3) y = x^3 + a_2 x^2 + a_4 x + a_6$$

$$E : y^2 + \underbrace{(a_1 x + a_3)}_{h(x)} y = x^3 + a_2 x^2 + a_4 x + a_6$$

Elliptic curves

$$E : y^2 + \underbrace{(a_1 x + a_3)}_{h(x)} y = \underbrace{x^3 + a_2 x^2 + a_4 x + a_6}_{f(x)}$$

Elliptic curves

$$E : y^2 + \underbrace{(a_1 x + a_3)}_{h(x)} y = \underbrace{x^3 + a_2 x^2 + a_4 x + a_6}_{f(x)} \quad , \qquad h,\, f \in \mathbb{F}_q[x]$$

usually $q = 2^r$ or $q = p$, prime.

$$E : y^2 + \underbrace{(a_1 x + a_3)}_{h(x)} y = \underbrace{x^3 + a_2 x^2 + a_4 x + a_6}_{f(x)} \ , \quad h,\, f \in \mathbb{F}_q[x]$$

usually $q = 2^r$ or $q = p$, prime.

$$E(\mathbb{F}_q) = \{\, (x,y) \in \mathbb{F}_q^2 : y^2 + h(x)y = f(x) \,\} \cup \{\, \infty \,\}$$

$$E\ :\ y^2 + \underbrace{(a_1 x + a_3)}_{h(x)} y = \underbrace{x^3 + a_2 x^2 + a_4 x + a_6}_{f(x)}\ , \qquad h,\ f \in \mathbb{F}_q[x]$$

usually $q = 2^r$ or $q = p$, prime.

$$E(\mathbb{F}_q) = \{\ (x,y) \in \mathbb{F}_q^2 : y^2 + h(x)y = f(x)\ \} \cup \{\ \infty\ \}$$

Commutative algebraic group with ∞ as zero element.

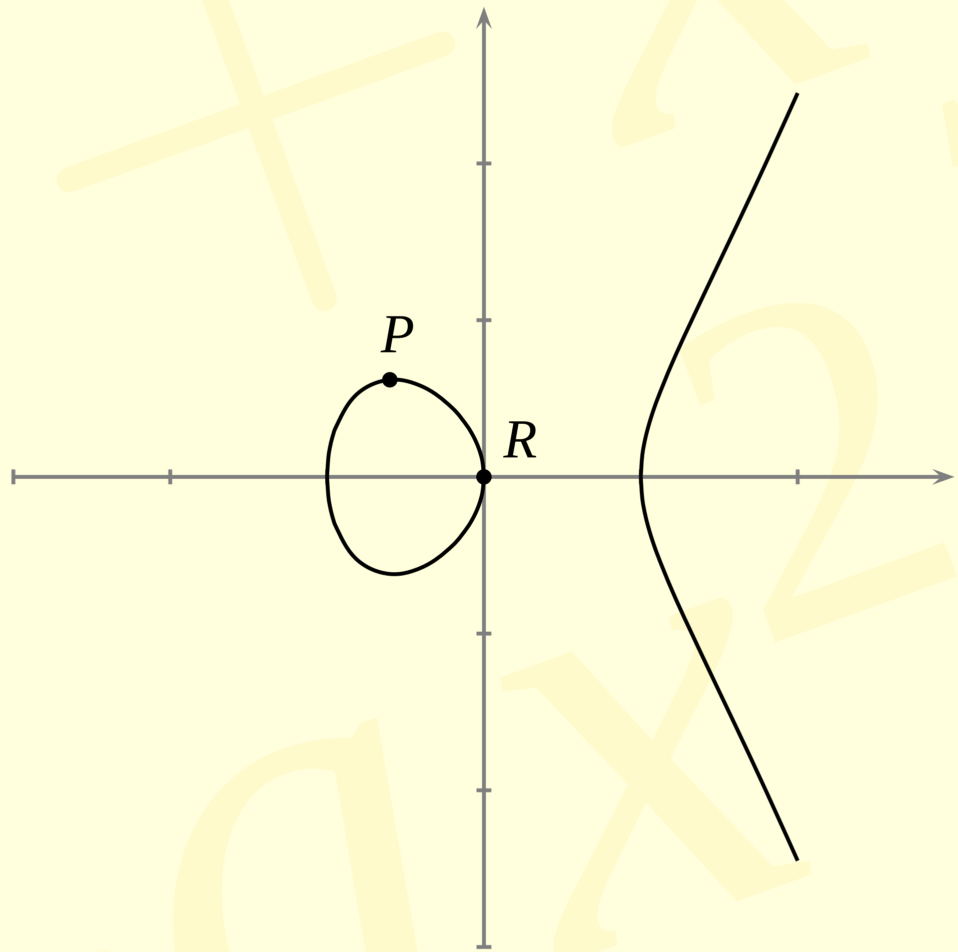
$P_1 = (x_1, y_1)\ \Rightarrow\ -P_1 = (x_1, -y_1 - a_1 x_1 - a_3)$.

Let $P_2 = (x_2, y_2)$. Then $P_3 = (x_3, y_3) = P_1 + P_2$ is given by

$$\begin{cases} x_3 &= -x_1 - x_2 - a_2 + \lambda(\lambda + a_1) \\ y_3 &= -y_1 - a_3 - a_1 x_3 + \lambda(x_1 - x_3) \end{cases} \quad \text{with} \quad \lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{if } P_1 \neq P_2, \\[2ex] \dfrac{3x_1^2 + 2a_2 x_1 + a_4 - a_1 y_1}{2y_1 + a_1 x_1 + a_3} & \text{if } P_1 = P_2. \end{cases}$$

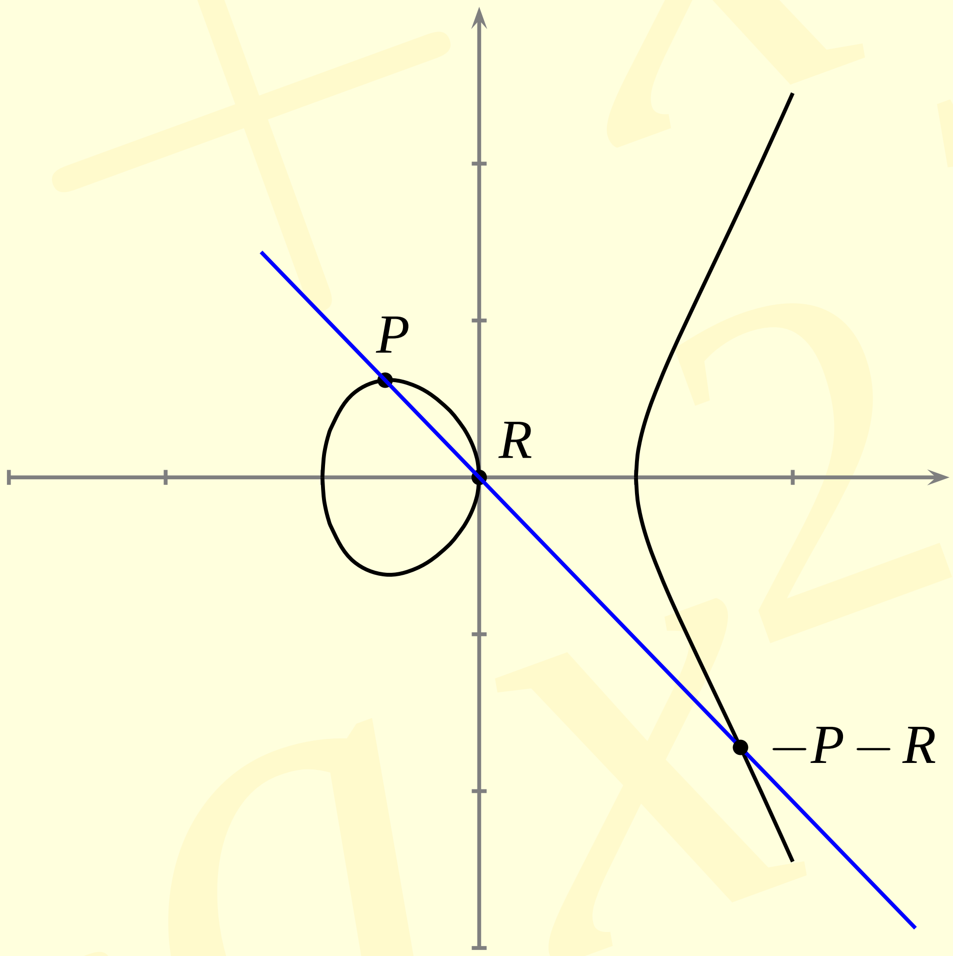
Elliptic Curves: ***Group Law in*** $E(\mathbb{R})$

$$E : y^2 = x^3 - x$$



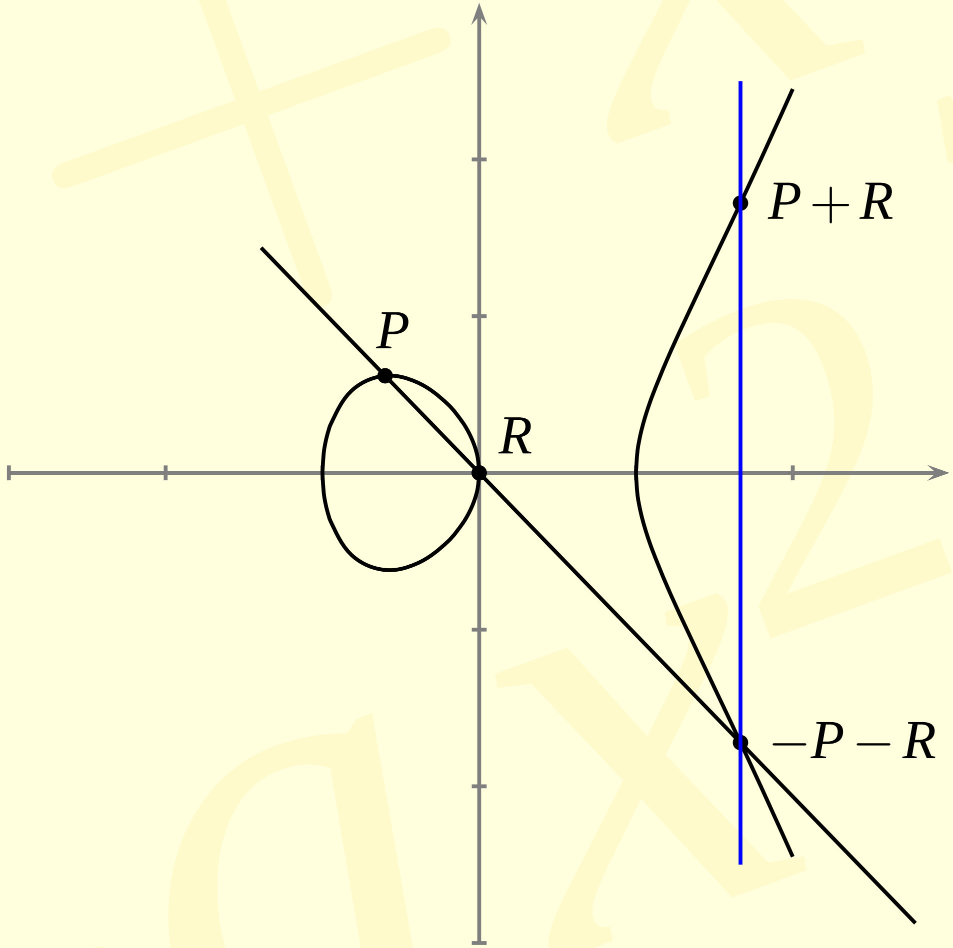
Elliptic Curves: ***Group Law in*** $E(\mathbb{R})$

$E : y^2 = x^3 - x$



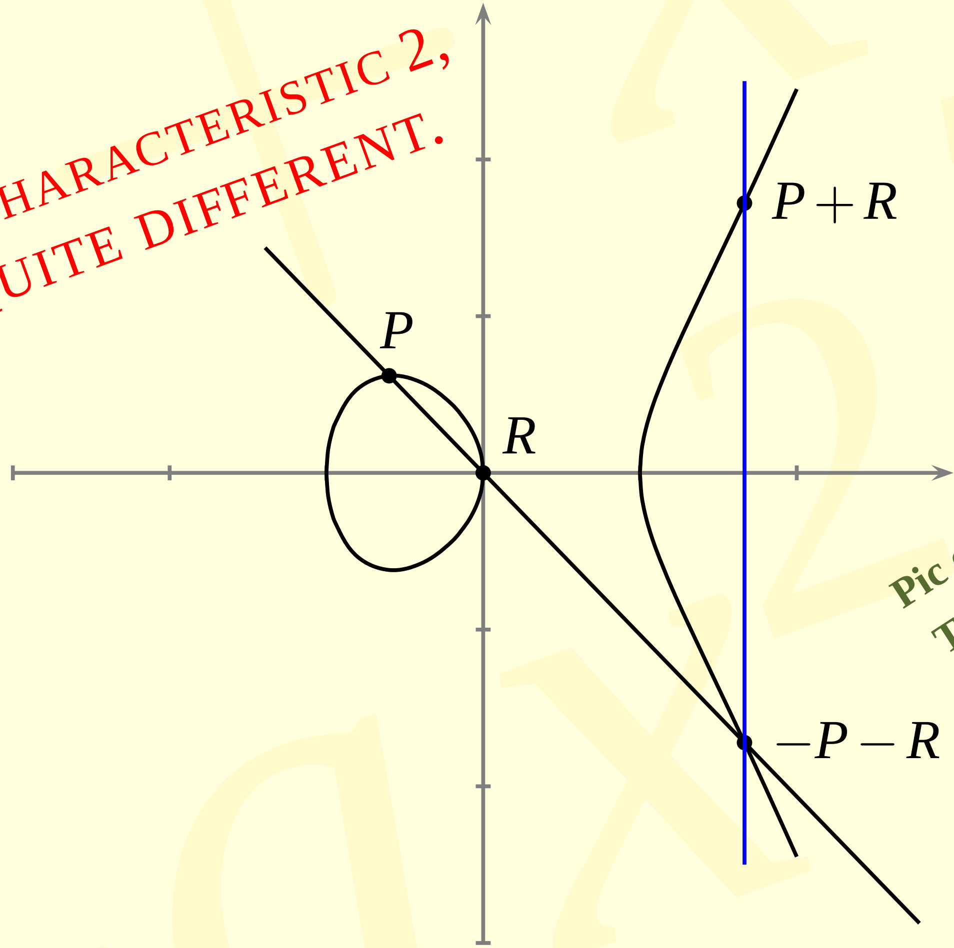
Elliptic Curves: Group Law in $E(\mathbb{R})$

$$E : y^2 = x^3 - x$$



Elliptic Curves: Group Law in $E(\mathbb{R})$

$$E : y^2 = x^3 - x$$



OF COURSE, IN CHARACTERISTIC 2, THIS LOOKS QUITE DIFFERENT.

Pic courtesy of Francesco, Mathieu and Tanja. Marc Joye will also use it soon!

Koblitz Curves

Special classes of elliptic curves have arithmetic properties that give good performance.

Koblitz Curves

Special classes of elliptic curves have arithmetic properties that give good performance.

Example: KOBLITZ CURVES.

Special classes of elliptic curves have arithmetic properties that give good performance.

Example: KOBLITZ CURVES.

(Solinas called them ABC curves.)

Special classes of elliptic curves have arithmetic properties that give good performance.

Example: Koblitz curves.

(Solinas called them ABC curves.)

Defined over $\mathbb{F}_{2^n}$ by equations of the form

$$E_a \,:\, y^2 + xy = x^3 + ax^2 + 1 \quad , \quad \text{with } \ a \in \{0,1\}.$$

Special classes of elliptic curves have arithmetic properties that give good performance.

Example: KOBLITZ CURVES.

(Solinas called them ABC curves.)

Defined over $\mathbb{F}_{2^n}$ by equations of the form

$$E_a \ : \ y^2 + xy = x^3 + a x^2 + 1 \ , \ \text{ with } \ a \in \{0, 1\}.$$

Why are they good?

Special classes of elliptic curves have arithmetic properties that give good performance.

Example: KOBLITZ CURVES.

(Solinas called them ABC curves.)

Defined over $\mathbb{F}_{2^n}$ by equations of the form

$$E_a \; : \; y^2 + xy = x^3 + a x^2 + 1 \;\; , \;\; \text{with} \;\; a \in \{0, 1\}.$$

Why are they good?

- Easy point counting. *(We are not doing this here.)*
- Fast arithmetic. *(We are doing this here.)*

Interlude: double-and-add

Want $s \cdot \boldsymbol{P}$: Write $s = \sum_{j=0}^{n-1} s_j 2^j$. Observe

$$s\boldsymbol{P} = 2(2(\cdots 2(2(s_{n-1}\boldsymbol{P}) + s_{n-2}\boldsymbol{P}) + \cdots) + s_1\boldsymbol{P}) + s_0\boldsymbol{P}$$

$\Rightarrow$ *double-and-add* algorithm (very old).

Interlude: *double-and-add*

Want $s \cdot \boldsymbol{P}$: Write $s = \sum_{j=0}^{n-1} s_j 2^j$. Observe

$$s\boldsymbol{P} = 2(2(\cdots 2(2(s_{n-1}\boldsymbol{P}) + s_{n-2}\boldsymbol{P}) + \cdots) + s_1\boldsymbol{P}) + s_0\boldsymbol{P}$$

$\Rightarrow$ *double-and-add* algorithm (very old).

We often have $s_j \in \{0, 1\}$. Other coefficients are possible: for example in the NAF $s_j \in \{0, \pm 1\}$ and $s_j s_{j+1} = 0$.

Interlude: double-and-add

Want $s \cdot \boldsymbol{P}$: Write $s = \sum_{j=0}^{n-1} s_j 2^j$. Observe

$$s\boldsymbol{P} = 2(2(\cdots 2(2(s_{n-1}\boldsymbol{P}) + s_{n-2}\boldsymbol{P}) + \cdots) + s_1\boldsymbol{P}) + s_0\boldsymbol{P}$$

$\Rightarrow$ *double-and-add* algorithm (very old).

We often have $s_j \in \{0,1\}$. Other coefficients are possible: for example in the NAF $s_j \in \{0,\pm 1\}$ and $s_j s_{j+1} = 0$.

If $\subseteq \{0,\pm 1\}$ and inversion of elements fast, the method is attractive for smart-cards.
(Reason: minimal memory requirements.)

Koblitz Curves: Here comes the Frobenius

$$E_a \; : \; y^2 + xy = x^3 + a x^2 + 1 \quad , \quad \text{with} \;\; a \in \{0,1\}.$$

- τ = *the Frobenius map* $\tau(x,y) = \left(x^2, y^2\right)$.

Koblitz Curves: ***Here comes the Frobenius***

$$E_a \ : \ y^2 + xy = x^3 + a x^2 + 1 \ , \ \text{ with } \ a \in \{0, 1\}.$$

- $\tau \ = \ $ *the Frobenius map* $\ \tau(x, y) = \left(x^2, y^2\right).$
- Using the addition formulæ easy to check that
  $$2(x, y) = (-1)^{1-a}\left(x^2, y^2\right) - \left(x^4, y^4\right)$$
  for all $(x, y) \in E_a$, i.e.:
- $2 = \mu\tau - \tau^2$ where $\mu = (-1)^{1-a}$ on E_a.

Koblitz Curves: ***I got τ … and now?***

Identify τ with a complex number satisfying

$$2 = \mu\tau - \tau^2 \quad , \quad \text{say} \quad \tau = \frac{\mu + \sqrt{-7}}{2}$$

We see then $\tau(\boldsymbol{P})$ as *multiplication of* $\boldsymbol{P}$ *by* τ.

(See? *complex* multiplication!)

We can multiply any point $\boldsymbol{P}$ by an element of $\mathbb{Z}[\tau]$.

Koblitz Curves: ***I got τ ... and now?***

Identify τ with a complex number satisfying

$$2 = \mu\tau - \tau^2 \quad , \qquad \text{say} \quad \tau = \frac{\mu + \sqrt{-7}}{2}$$

We see then $\tau(\boldsymbol{P})$ as *multiplication of* $\boldsymbol{P}$ *by* τ.

(See? *complex* multiplication!)

We can multiply any point $\boldsymbol{P}$ by an element of $\mathbb{Z}[\tau]$.

τ-adic non-adjacent form (τ-NAF) associated to $s \in \mathbb{Z}[\tau]$:

$$s = \sum_i s_i \tau^i \quad \text{with} \quad s_j s_{j+1} = 0.$$

In particular $\sum_{i=0}^{m} s_i \tau^i(\boldsymbol{P}) = s\boldsymbol{P}$ for all $\boldsymbol{P} \in E_a(\mathbb{F}_{2^n})$.

$\Rightarrow$ use τ-and-add instead of double-and-add.

Koblitz Curves: Rounding off

τ is *very fast*. Using it in place of doubling, and the τ-NAF in place of a NAF, makes scalar multiplication fast ...

Koblitz Curves: Rounding off

τ is *very fast*. Using it in place of doubling, and the τ-NAF in place of a NAF, makes scalar multiplication fast ...

... if the τ-adic expansion is not too long and not too dense.

Koblitz Curves: Rounding off

τ is *very fast*. Using it in place of doubling, and the τ-NAF in place of a NAF, makes scalar multiplication fast ...

... if the τ-adic expansion is not too long and not too dense.

In fact, length is $\log_2 N_{\mathbb{Q}(\tau)/\mathbb{Q}}(s) \approx 2n$ and density $\frac{1}{3}$.
($\frac{2}{3}n$ adds for one scalar product instead of $\frac{4}{3}n$)

Koblitz Curves: Rounding off

τ is *very fast*. Using it in place of doubling, and the τ-NAF in place of a NAF, makes scalar multiplication fast ...

... if the τ-adic expansion is not too long and not too dense.

In fact, length is $\log_2 N_{\mathbb{Q}(\tau)/\mathbb{Q}}(s) \approx 2n$ and density $\frac{1}{3}$.
($\frac{2}{3}n$ adds for one scalar product instead of $\frac{4}{3}n$)

But Solinas showed how to make it shorter:

- First attempt: Reduce s by $\tau^n - 1$. Problem: slow.

Koblitz Curves: Rounding off

τ is *very fast*. Using it in place of doubling, and the τ-NAF in place of a NAF, makes scalar multiplication fast ...

... if the τ-adic expansion is not too long and not too dense.

In fact, length is $\log_2 N_{\mathbb{Q}(\tau)/\mathbb{Q}}(s) \approx 2n$ and density $\frac{1}{3}$.
($\frac{2}{3}n$ adds for one scalar product instead of $\frac{4}{3}n$)

But Solinas showed how to make it shorter:

- First attempt: Reduce s by $\tau^n - 1$. Problem: slow.
- Solution: Use slightly longer expansion.
  Length $\ell \leqslant n + a + 3$, but reduction time negligible.

Point Halving

E.W. Knudsen and R. Schroeppel had a *funny* idea for *generic elliptic curves over fields of characteristic two*.

Instead of doubling points, they thought of *halving* them.

If $\boldsymbol{P} \in E(\mathbb{F}_{2^n})$ *is a point of large prime order q, find* $\boldsymbol{R}$ *(also of order q) such that* $2\boldsymbol{R} = \boldsymbol{P}$.

If the idea can be realized, one can turn the scalar upside-down and do a halve-and-add in place of the double-and-add method.

If halving faster than doubling, then idea useful.

Point Halving: How to do it – 1

E = elliptic curve over $\mathbb{F}_{2^n}$

$$E \ : \ y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in \mathbb{F}_{2^n}$ and $G \leqslant E(\mathbb{F}_{2^n})$ of large prime order.

If $\boldsymbol{P} = (x, y)$ define $\lambda_{\boldsymbol{P}} = x + \dfrac{y}{x}$.

Let $\boldsymbol{P} = (x, y)$, $\boldsymbol{R} = (u, v) \in E(\mathbb{F}_{2^n}) \setminus \{0\}$ with $2\boldsymbol{R} = \boldsymbol{P}$.
Then

$$\lambda_{\boldsymbol{R}} = u + \frac{v}{u} \tag{1}$$

$$x = \lambda_{\boldsymbol{R}}^2 + \lambda_{\boldsymbol{R}} + a \tag{2}$$

$$y = u^2 + x(\lambda_{\boldsymbol{R}} + 1) \tag{3}$$

Point Halving: ***How to do it – 2***

A reminder:

Let $\boldsymbol{P} = (x,y)$, $\boldsymbol{R} = (u,v) \in E(\mathbb{F}_{2^n}) \setminus \{0\}$ with $2\boldsymbol{R} = \boldsymbol{P}$.

$$\lambda_{\boldsymbol{R}} = u + \frac{v}{u} \tag{1}$$

$$x = \lambda_{\boldsymbol{R}}^2 + \lambda_{\boldsymbol{R}} + a \tag{2}$$

$$y = u^2 + x(\lambda_{\boldsymbol{R}} + 1) \tag{3}$$

Given $\boldsymbol{P}$, *point halving* consists in finding $\boldsymbol{R}$.

Point Halving: *How to do it* – 2

A reminder:

Let $\boldsymbol{P} = (x,y)$, $\boldsymbol{R} = (u,v) \in E(\mathbb{F}_{2^n}) \setminus \{0\}$ with $2\boldsymbol{R} = \boldsymbol{P}$.

$$\lambda_{\boldsymbol{R}} = u + \frac{v}{u} \tag{1}$$

$$x = \lambda_{\boldsymbol{R}}^2 + \lambda_{\boldsymbol{R}} + a \tag{2}$$

$$y = u^2 + x(\lambda_{\boldsymbol{R}} + 1) \tag{3}$$

Given $\boldsymbol{P}$, *point halving* consists in finding $\boldsymbol{R}$. $\Leftrightarrow$

$\Leftrightarrow$ Solve (2) for $\lambda_{\boldsymbol{R}}$, (3) for u, and finally (1) for v. $\Leftrightarrow$

(i) Solve $\lambda_{\boldsymbol{R}}^2 + \lambda_{\boldsymbol{R}} = a + x$ for $\lambda_{\boldsymbol{R}}$

(ii) Put $t = y + x(\lambda_{\boldsymbol{R}} + 1)$

(iii) Find u with $u^2 = t$

(iv) Put $v = t + u\lambda_{\boldsymbol{R}}$.

Let $\boldsymbol{P} = (x,y)$, $\boldsymbol{R} = (u,v) \in E(\mathbb{F}_{2^n}) \setminus \{0\}$ with $2\boldsymbol{R} = \boldsymbol{P}$.
Let $\#E(\mathbb{F}_{2^n}) = 2q$. If $\boldsymbol{P}$ has order q, want $\boldsymbol{R}$ also of order q

$$\begin{array}{rl} \text{(i)} & \text{Solve } \lambda_{\boldsymbol{R}}^2 + \lambda_{\boldsymbol{R}} = a + x \text{ for } \lambda_{\boldsymbol{R}} \\ \text{(ii)} & \text{Put } t = y + x(\lambda_{\boldsymbol{R}} + 1) \\ \text{(iii)} & \text{Find } u \text{ with } u^2 = t \\ \text{(iv)} & \text{Put } v = t + u\lambda_{\boldsymbol{R}} \quad . \end{array}$$

Yields 2 points $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, one of order q and the other $2q$ ($\boldsymbol{R}_1 - \boldsymbol{R}_2$ has order 2) $\Leftrightarrow$ the 2 solutions of (i).

Solution: attempt another doubling – indeed, right after (i).
If successful, $\boldsymbol{R}$ has order q.
If not, it must have order $2q$: Replace $\lambda_{\boldsymbol{R}}$ by $\lambda_{\boldsymbol{R}} + 1$.

Point Halving: Does it work? Yes!

M = cost of a field multiplication.
Knudsen and Schroeppel (and Fong, Hankerson, Lopez and Menezes) show that:

- Extracting square roots costs like a squaring ($\frac{1}{2}M$ or 0).
- Solving $\lambda^2 + \lambda = c$ costs $\frac{2}{3}M$.

Now:

- Point addition $= 1I + 2M + 1S$. $1I \approx 8$–$10M$.
- Point doubling $= 1I + 2M + 1S$.
- Point halving $= 2M +$ equation $+ \sqrt{}$ + extra cost.

Extra cost = 0 if E has minimal 2-torsion. Otherwise bigger.

$\Rightarrow$ *for many curves, using point halving wins big (cit.).*

Since point halving is slower than a Frobenius operation, it is going to be of no use for speeding up scalar multiplication on Koblitz curves.

Indeed, halve-and-add is slower than τ-and-add.

But this is not the whole story.

If you can use both, you indeed win bigger.

Simplifying τ-adic expressions: An observation

$$E_a \ : \ y^2 + xy = x^3 + ax^2 + 1 \ \ , \ \ \text{with} \ \ a \in \{0,1\}.$$

$$2 = \mu\tau - \tau^2 \quad \text{where} \ \ \mu = (-1)^{1-a} \ \ \text{on} \ \ E_a$$

from which

$$2 = -\mu\left(\tau^2 + 1\right)\tau \ \ .$$

In other words, if $\boldsymbol{P} = 2\boldsymbol{R}$ and $\boldsymbol{Q} = \tau\boldsymbol{R}$, then:

$$2\boldsymbol{R} = -\mu\left(\tau^2 + 1\right)\tau\boldsymbol{R} \ \ ,$$

or

$$\boldsymbol{P} = -\mu\left(\tau^2 + 1\right)\boldsymbol{Q} \ \ .$$

Use telescopic sums!

Simplifying τ-adic expressions: An observation

$$E_a \;:\; y^2 + xy = x^3 + ax^2 + 1 \;\;,\;\; \text{with} \;\; a \in \{0,1\}.$$

$$2 = \mu\tau - \tau^2 \quad \text{where} \;\; \mu = (-1)^{1-a} \;\; \text{on} \;\; E_a$$

Notation: $\langle \ldots s_j s_{j-1} \ldots s_1 s_0 \rangle_\tau = \sum s_j \tau^j$ as with binary expansions of integers.

Using telescopic sums, more sequences follow...

$$\langle 10\bar{1}01 \rangle_\tau P = \langle 100001 \rangle_\tau Q$$

$$\langle 10101 \rangle_\tau P = \langle 10\bar{1} \rangle_\tau Q$$

Recall:
$P = 2R$ and
$Q = \tau R$.

or even

$$\langle 1010101\bar{0}1 \rangle_\tau P = \langle 1000000\bar{1} \rangle_\tau Q \quad .$$

in the case $a = 1$, hence $\mu = 1$.

Simplifying τ-adic expressions: An observation

The following expressions have something in common:

$$\langle 10\bar{1}01\rangle_\tau P = \langle 100001\rangle_\tau Q$$
$$\langle 10101\rangle_\tau P = \langle 10\bar{1}\rangle_\tau Q$$

or even

$$\langle 101010\bar{1}01\rangle_\tau P = \langle 1000000\bar{1}\rangle_\tau Q \quad .$$

The left hand sides are portions of τ-adic NAFs, with (highest possible) density 1/2.
The expressions on the right hand side represent the same element of $E_a(\mathbb{F}_{2^n})$ but the "scalar" has just weight 2.
*Such sequences are called k-*blocks*. k = # of nonzeros.*

Simplifying τ-adic expressions: An observation

The following expressions have something in common:

$$\langle 1 0 \bar{1} 0 1 \rangle_\tau \boldsymbol{P} = \langle 1 0 0 0 0 1 \rangle_\tau \boldsymbol{Q}$$

$$\langle 1 0 1 0 1 \rangle_\tau \boldsymbol{P} = \langle 1 0 \bar{1} \rangle_\tau \boldsymbol{Q}$$

or even

$$\langle 1 0 1 0 1 0 \bar{1} 0 1 \rangle_\tau \boldsymbol{P} = \langle 1 0 0 0 0 0 0 \bar{1} \rangle_\tau \boldsymbol{Q} \quad .$$

But, there's more: *There are three infinite families of τ-adic expressions $\mathcal{S}$ of density 1/2, with the property that $\mathcal{S}\boldsymbol{P} = \mathcal{S}'\boldsymbol{Q}$ for a suitable τ-adic expression $\mathcal{S}'$ of weight 2.*
The sequences that simplify are called good *k-blocks.*

Simplifying τ-adic expressions: The general result

(ωriginal times P) $\quad \omega_i^k P = \rho_i^k Q \quad$ (ρeplacement times Q)

Expressed as sequences:

(Go to complexity)

$$\langle \underbrace{\bar{1}^{k-1}\,0\,\bar{1}^{k-2}\,0 \quad \ldots \quad 0\,1\,0\,\bar{1}\,0\,1}_{\text{length } 2k-1} \rangle P = \bar{\mu} \langle \underbrace{\bar{1}^{k-1}\,0\,0 \ \ldots \ 0\,0\,1}_{\text{length } 2k+1} \rangle Q \quad (i = 1)$$

$$\langle \underbrace{\bar{1}^{k-2}\,0\,\bar{1}^{k-2}\,0\,\bar{1}^{k-3}\,0 \quad \ldots \quad 0\,1\,0\,\bar{1}\,0\,1}_{\text{length } 2k-1} \rangle P = \langle \underbrace{\bar{1}^{k-1}\,0\,0 \ldots 0\,0\,\bar{\mu}}_{\text{length } 2k} \rangle Q \quad (i = 2)$$

$$\langle \underbrace{\bar{1}^{k-3}\,0\,\bar{1}^{k-3}\,0\,\bar{1}^{k-3}\,0\,\bar{1}^{k-4}\,0 \ldots 0\,1\,0\,\bar{1}\,0\,1}_{\text{length } 2k-1} \rangle P = \langle \underbrace{\bar{1}^{k-3}\,0\,0 \ldots 0\,\bar{\mu}}_{\text{length } 2k-2} \rangle Q \quad (i = 3)$$

How to use these equalities to speed-up scalar multiplication?

From the τ-NAF $\mathcal{S}$ of s, create *two* τ-adic expansions, $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$, by replacing subsequences, where:

1. $\mathcal{S}^{(1)}$ is obtained from $\mathcal{S}$ by removing the **o**riginal sequences that admit simplifications
2. $\mathcal{S}^{(2)}$ consists of the weight 2 **r**eplacements of the sequences removed from $\mathcal{S}$, each at the same position where the original subsequence was in $\mathcal{S}$.

If other words, for each $\pm\omega_i^k\tau^j$ subtracted from $\mathcal{S}$ to build $\mathcal{S}^{(1)}$, the sequence $\pm\rho_i^k\tau^j$ is added to $\mathcal{S}^{(2)}$.

Since $\omega_i^k P = \rho_i^k Q$ we have: $sP = \mathcal{S}^{(1)}P + \mathcal{S}^{(2)}Q$.

The new recoding: The algorithm

The algorithm processes the input τ-NAF from left to right. I.e. from the coefficients of the lower powers of τ.

0. Zeros are skipped ...
1. ... until a 1 or $\bar{1}$ is found, the first “bit” in a block. The following zero is skipped.
2. Then a series of bits of alternating signs is read (with single zeros in between) – and added to the block.

3, 4. And at most two bits of the same sign of the previous one are read, and put in the block.

$$\ldots 0\,0 \,\langle\, \bar{1}^{k-3}\, 0\, \bar{1}^{k-3}\, 0\, \bar{1}^{k-3}\, 0\, \bar{1}^{k-4}\, 0 \;\ldots\; 0\,1\,0\,\bar{1}\,0\,1 \,\rangle\, 0\,0 \ldots$$

The new scalar product: The Normal Basis case

If the field $\mathbb{F}_{2^n}$ is represented via a normal basis, squarings are free.

We do not need double scalar multiplication to compute $\mathcal{S}^{(1)}\boldsymbol{P} + \mathcal{S}^{(2)}\boldsymbol{Q}$ and we do not even need to store $\boldsymbol{Q}$.
We do instead the following:

- First compute $\mathcal{S}^{(2)}\boldsymbol{P}$.
- Halve the result and apply τ.
- Resume the τ-and-add loop using $\mathcal{S}^{(1)}$.

The new scalar product: The Normal Basis case

If the field $\mathbb{F}_{2^n}$ is represented via a normal basis, squarings are free.

We do not need double scalar multiplication to compute $\mathcal{S}^{(1)}\boldsymbol{P} + \mathcal{S}^{(2)}\boldsymbol{Q}$ and we do not even need to store $\boldsymbol{Q}$.
We do instead the following:

- First compute $\mathcal{S}^{(2)}\boldsymbol{P}$.
- Halve the result and apply τ.
- Resume the τ-and-add loop using $\mathcal{S}^{(1)}$.

We double the Frobenius operations: Does not matter!

We also interleave with the recoding of $\mathcal{S}$ into $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ to have an algorithm without additional memory requirements, apart from code and a few variables.

To compute the complexity of the algorithm...

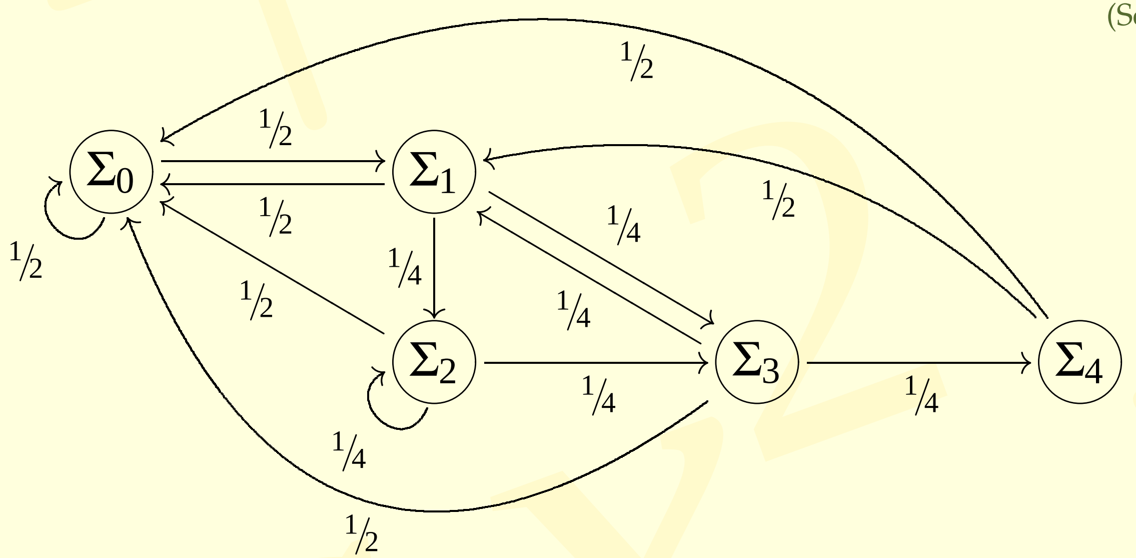
... is to compute # non-zero coefficients in $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$.

$\mathcal{S}$ contains about $\frac{1}{3}(n+a+3)$ of them.

We describe the recoding algorithm as a Markov chain:

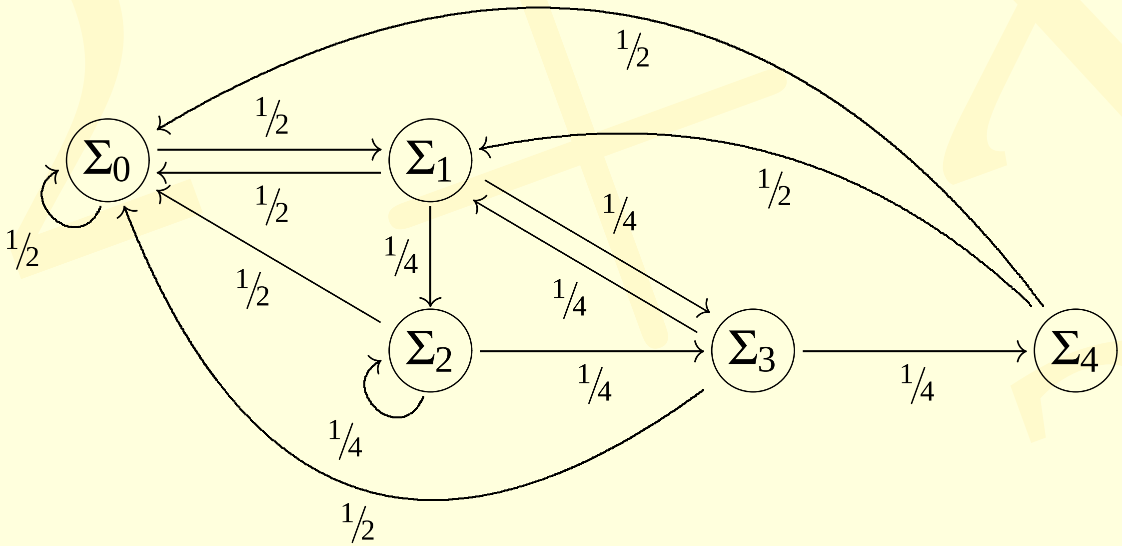
(See all sequences again)

(Skip next slide)



and get that $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ have about $\frac{2}{7}(n+a+3)$ non-zero coefficients. $\left(\frac{1}{3}-\frac{2}{7}\right)/\frac{1}{3} \approx 14.29\%$ less than the τ-NAF!

To compute the complexity of the algorithm...



The states:

Σ_0: Zeros between

Σ_1: First bit (lsb)

Σ_2: Alternating signs

Σ_3: 1rst equal sign

Σ_4: 2nd equal sign

$$\langle\, \bar{1}^{k-1}\, 0\, \bar{1}^{k-2}\, 0 \quad \ldots \quad 0\,1\,0\,\bar{1}\,0\,1 \,\rangle \cdot \boldsymbol{P}$$

$$\langle\, \bar{1}^{k-2}\, 0\, \bar{1}^{k-2}\, 0\, \bar{1}^{k-3}\, 0 \quad \ldots \quad 0\,1\,0\,\bar{1}\,0\,1 \,\rangle \cdot \boldsymbol{P}$$

$$\langle\, \bar{1}^{k-3}\, 0\, \bar{1}^{k-3}\, 0\, \bar{1}^{k-3}\, 0\, \bar{1}^{k-4}\, 0 \quad \ldots \quad 0\,1\,0\,\bar{1}\,0\,1 \,\rangle \cdot \boldsymbol{P}$$

Open Problems

- Usage of more point halvings?
  - For now, little or no improvement found.

Open Problems

- Usage of more point halvings?
  - For now, little or no improvement found.
- Combine this trick with width-w τ-NAF?
- Hyperelliptic Koblitz curves?
  - Width-w τ-NAF and HEC's $\Rightarrow$ same problem: larger coefficient sets. It is not obvious how to simplify those τ-adic expansions. Or maybe we are just lazy cuz there are too many of them ;-)

- Usage of more point halvings?
  - For now, little or no improvement found.
- Combine this trick with width-w τ-NAF?
- Hyperelliptic Koblitz curves?
  - Width-w τ-NAF and HEC's $\Rightarrow$ same problem: larger coefficient sets. It is not obvious how to simplify those τ-adic expansions. Or maybe we are just lazy cuz there are too many of them ;-)
- Our method works for elliptic curves, but there are other genus one objects which are of great interest for the whole cryptographic community. Especially during cold winters ...

Elliptic socks!

Photo by Jean-Jacques Quisquater. Socks made by Tanja Lange for Mathieu Ciet.

Conclusions

First combination of Point Halving with Frobenius and τ-adic expansions.

- New scalar decomposition – $\mathcal{S}\boldsymbol{P} = \mathcal{S}^{(1)}\boldsymbol{P} + \mathcal{S}^{(2)}\boldsymbol{Q}$ with $\boldsymbol{Q} = \tau(\boldsymbol{P}/2)$ – with $\approx$ 14.29% less non-zero coeffs than the τ-NAF $\mathcal{S}$.
- If normal bases used (in HW) $\approx$ 14.29% less group ops.
- In software implementations expect 8.7 to 12% speed-up for 163 and 233 bit curves.
- No additional memory requirements (surprise) apart from code and some vars (no precomputed pts!).
  $\Rightarrow$ can be used where the old τ-NAF is used.

References

- J. A. SOLINAS. *Efficient Arithmetic on Koblitz Curves*. Designs, Codes and Cryptography, Vol. 19 (2000), No. 2/3, pp. 125–179.

References

- J. A. SOLINAS. *Efficient Arithmetic on Koblitz Curves*. Designs, Codes and Cryptography, Vol. 19 (2000), No. 2/3, pp. 125–179.
- E. W. KNUDSEN. *Elliptic Scalar Multiplication Using Point Halving*. In: *Advances in Cryptography - ASIACRYPT 1999*, LNCS 1716, pp. 135–149. Springer, 1999.

References

- J. A. SOLINAS. *Efficient Arithmetic on Koblitz Curves*. Designs, Codes and Cryptography, Vol. 19 (2000), No. 2/3, pp. 125–179.
- E. W. KNUDSEN. *Elliptic Scalar Multiplication Using Point Halving*. In: *Advances in Cryptography - ASIACRYPT 1999*, LNCS 1716, pp. 135–149. Springer, 1999.
- K. FONG, D. HANKERSON, J. LOPEZ AND A. MENEZES. *Field inversion and point halving revisited*. Available from `http://www.cs.siu.edu/~kfong/research/ECCpaper.ps`

References

- J. A. SOLINAS. *Efficient Arithmetic on Koblitz Curves*. Designs, Codes and Cryptography, Vol. 19 (2000), No. 2/3, pp. 125–179.
- E. W. KNUDSEN. *Elliptic Scalar Multiplication Using Point Halving*. In: *Advances in Cryptography - ASIACRYPT 1999*, LNCS 1716, pp. 135–149. Springer, 1999.
- K. FONG, D. HANKERSON, J. LOPEZ AND A. MENEZES. *Field inversion and point halving revisited*. Available from `http://www.cs.siu.edu/~kfong/research/ECCpaper.ps`
- R. AVANZI, M. CIET AND F. SICA. *Faster Scalar Multiplication on Koblitz Curves combining Point Halving with the Frobenius Endomorphism*. Preprint.

References

- J. A. SOLINAS. *Efficient Arithmetic on Koblitz Curves*. Designs, Codes and Cryptography, Vol. 19 (2000), No. 2/3, pp. 125–179.
- E. W. KNUDSEN. *Elliptic Scalar Multiplication Using Point Halving*. In: *Advances in Cryptography - ASIACRYPT 1999*, LNCS 1716, pp. 135–149. Springer, 1999.
- K. FONG, D. HANKERSON, J. LOPEZ AND A. MENEZES. *Field inversion and point halving revisited*. Available from `http://www.cs.siu.edu/~kfong/research/ECCpaper.ps`
- R. AVANZI, M. CIET AND F. SICA. *Faster Scalar Multiplication on Koblitz Curves combining Point Halving with the Frobenius Endomorphism*. Preprint.
- T. LANGE. *Applications of Knitting to Cryptology.* Work always in progress (maybe even as I speak).

(m)Any questions?

? ? ? ? ?