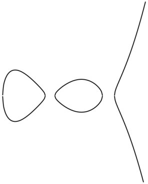
The state-of-the-art in hyperelliptic curve cryptography

Craig Costello

Workshop on Curves and Applications
Calgary, Canada
August 19, 2013

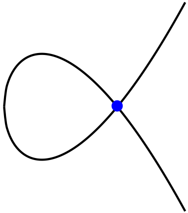


Microsoft Research

PIMS

Thanks for inviting/rescuing me. . .

- Thanks to Mark, Michael and Renate, I get to hear about . . .
  - *Counting Abelian Surfaces*
  - *Divisor Computations using Global Sections*
  - *Isogeny-Based Cryptography*
  - *Splitting of Abelian Varieties*
  - *Explicit Isogenies*



. . . instead of being at CRYPTO'13, and hearing about . . .

  - *Leakage-Resilient Symmetric Cryptography Under Empirically Verifiable Assumptions*
  - *Plain versus Randomized Cascading-Based Key-Length Extension for Block Ciphers*
  - *On the Achievability of Simulation-Based Security for Functional Encryption*
  - *. . . etc etc . . .*

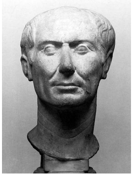
Outline

1. Motivation/overview/preliminaries
   - fast and compact public-key crypto
   - genus 1 vs. genus 2
   - the ECDLP and scalar multiplication
2. Genus 1 vs. Genus 2 (three fights)
   - `CurveP-256` **vs.** `generic1271`
   - `2GLV` **vs.** `4GLV`
   - `curve25519` **vs.** `Kummer1271`
3. Three open problems in genus 2
   - GLV on the Kummer surface?
   - Making genus 2 truly resistant
   - Faster arithmetic...

1. Motivation/overview/preliminaries

Private-key vs. Public-key cryptography

BC - WWII:



Caesar

Mary, Queen of Scots

German Enigma Code

must communicate before sharing secrets

1970's:

Diffie-Hellman-Merkle

Rivest-Shamir-Adleman (RSA)

Cocks

HUGE BREAKTHROUGH: no need for prior communication!!!

Diffie-Hellman (Merkle): a toy example

Public values:

$q = 1000000000000061$ (prime), $g = 832022676086941$ (generator of $\mathbb{Z}_q$).

Secret values:

Alice's secret: a=4275315603725493

Alice computes (public key):

$g^a \bmod q = 9213047582249495$

Bob can compute:

Bob's secret: b=1083333300180813

Bob computes (public key):

$g^b \bmod q = 9893308140872135$

Alice can compute:

$$9893308140872135^a = 8817060794020263 = 9213047582249495^b$$

$$= g^{ab}$$

Secret keys safe as long as discrete log problem (DLP) is hard

Joint secret safe as long as Diffie-Hellman problem is hard

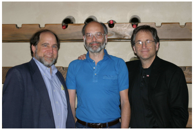
Modulus (key) sizes: then and now

1970's:



$q =$
1606938044258990275541962092341162602522202993782792835301301.
(200-bit prime)

NOW:


$q =$
5809605995369958062859502533304574370686975176362895236661486152287203730997110225737336044533118407251
3261577549805174439905295945400471216628856721870324010321116397064404988440498509890516272002447658070
4181239472968054002410482797658436938152229236120877904476989274322575173807697956881130957912551133309
3243519553784816306381580161860200247492568448150242515304449577187604136428738580990172551573934146255
8303664059150008696437320532185668325452911079037228316341385995864066903259597251874471690595408050123
1020963901175074876001709536073423494575741627299485601330861695852995830467763701918159408852834506128
5863898271763457294883546638879554311615446446330199254382340016292057090751175533888161918987295591531
5366987012922676854655174379157908231548446347802601028917180324953960750418994855138111269773074789690
748570437107161501213159220245567592412390131529197109564684063794429149416143571079144625673296936 49
(3072-bit prime)

Curves are much better than $\mathbb{F}_q^*$



'76

$\mathbb{F}_q^*$ (today $q \approx 3072$ bits)



'85

$E/\mathbb{F}_q$ (today $q \approx 256$ bits)


'89

$\mathrm{Jac}(C_g/\mathbb{F}_q)$ (today, $g = 2$, $q \approx 128$ bits)

Curves are much better than $\mathbb{F}_q^*$

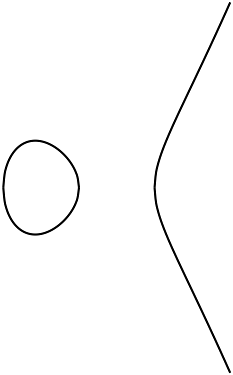
  '76 $\mathbb{F}_q^*$ (**BORING**)

  '85 $E/\mathbb{F}_q$ (**FUN**)

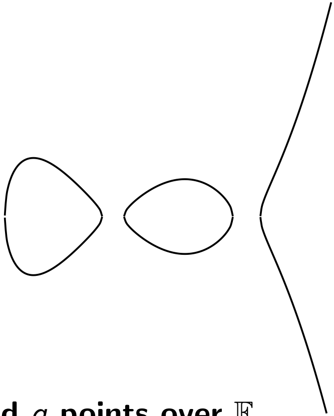
 '89 $\mathrm{Jac}(C_g/\mathbb{F}_q)$ (**FUNNER**)

Why fields of half the size?

$y^2 = x^3 + a_2x^2 + a_1x + a_0$



$y^2 = x^5 + b_4x^4 + \cdots + b_0$

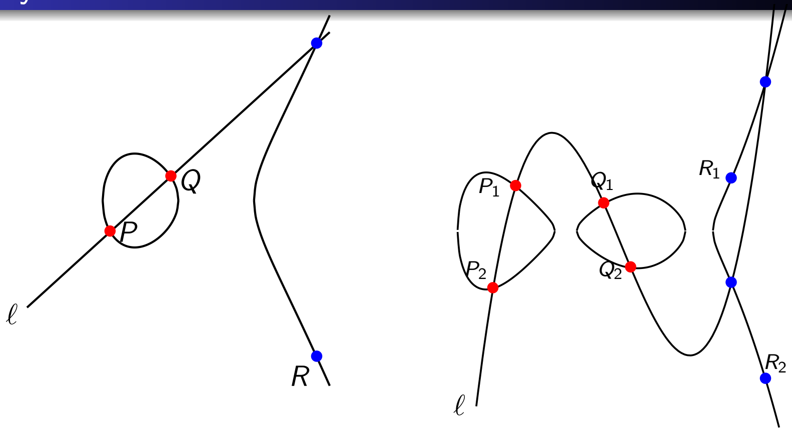


Both curves have around q points over $\mathbb{F}_q$

Hasse-Weil: $q + 1 - 2g\sqrt{q} \leq \#C(\mathbb{F}_q) \leq q + 1 + 2g\sqrt{q}$

$(g = \text{genus})$

Why fields of half the size?



Roughly speaking: group elements are pairs of points

$$\mathrm{Pic}^0_C = \mathrm{Div}^0_C / \mathrm{Prin}_C$$

Riemann-Roch: unique reduced rep. of "weight" at most g

$\#E(\mathbb{F}_q) \approx q$ **vs.** $\#\mathrm{Jac}(C)(\mathbb{F}_q) \approx q^2$

Hasse-Weil: $(q^{1/2} - 1)^{2g} \leq |\mathrm{Pic}^0_C| \leq (q^{1/2} + 1)^{2g}$

Three fights (over prime fields)

Genus 1 - elliptic		**Genus 2 - hyperelliptic**
`CurveP-256` (NIST) $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ $E/\mathbb{F}_p \colon y^2 = x^3 - 3x + b$ $\#E = r$ (256-bit prime)	VS. (generic)	`Generic1271` $p = 2^{127} - 1$ $C/\mathbb{F}_p \colon y^2 = x^5 + a_3x^3 + \cdots + a_0$ $\#\mathrm{Jac} = r$ (254-bit prime)
`GLV-j=0` (Longa-Sica) $p = 2^{256} - 11733$ $E/\mathbb{F}_p \colon y^2 = x^3 + 2$ $\#E = r$ (256-bit prime)	VS. (endos)	`BuhlerKoblitzGLV` $p = 2^{64} \cdot (2^{63} - 27443) + 1$ $C/\mathbb{F}_p \colon y^2 = x^5 + 17$ $\#\mathrm{Jac} = r$ (254-bit prime)
`curve25519` (Bernstein) $p = 2^{255} - 19$ $E/\mathbb{F}_p \colon y^2 = x^3 + 486662x^2 + x$ $\#E = 2^3 \cdot r$ (253-bit prime)	VS. (ladder)	`Kummer1271` $p = 2^{127} - 1$ $C/\mathbb{F}_p \colon y^2 = x^5 + a_3x^3 + \cdots + a_0$ $\#\mathrm{Jac} = 2^4 \cdot r$ (251-bit prime)

The discrete logarithm problem on Jacobians

The ECDLP or (H)ECDLP

Given $P, [n]P \in \mathrm{Jac}(C)$, find n.

- Here $[n]P = \underbrace{P + P + \cdots + P}_{n \text{ times}}$
- e.g. on `CurveP-256`, $[P, [n]P] =$
$[(40479349090799629115126637582848697209588271547831167017773909685338681225599,$
$22967748547577358811128749528539359233496570666630926906982292826073120749928),$
$74180245058659284846967422193612971784890177538113222391105953224411036727045,$
$110900663252159927273776818506962683131310742871875440526518883183068216925159)]$
- e.g. on `generic1271`, $[P, [n]P] =$
$[(x^2 + 75376293723959170227940456903550835710x + 135725164365695293093314509380448016967,$
$105339129574254139412560007100896944713x + 113195465952718396500669047047242028400),$
$x^2 + 119268206887311488578575035256786375387x + 158619788005039757255593506567270537230,$
$98156413785948877596533722507100341843x + 85481124418552453788443079432675460759)]$

The discrete logarithm problem on Jacobians

The ECDLP or (H)ECDLP

Given $P, [n]P \in \mathrm{Jac}(C)$, find n.

- Here $[n]P = \underbrace{P + P + \cdots + P}_{n \text{ times}}$
- e.g. on `CurveP-256`, $[P, [n]P] =$
  $$[(40479349090799629115126637582848697209588271547831167017773909685338681225599,$$
  $$22967748547577358811128749528539359233496570666630926906982292826073120749928),$$
  $$74180245058659284846967422193612971784890177538113222391105953224411036727045,$$
  $$110900663252159927273776818506962683131310742871875440526518883183068216925159)]$$
- e.g. on `generic1271`, $[P, [n]P] =$
  $$[(x^2 + 75376293723959170227940456903550835710x + 135725164365695293093314509380448016967,$$
  $$105339129574254139412560007100896944713x + 113195465952718396500669047047242028400),$$
  $$x^2 + 119268206887311488578575035256786375387x + 158619788005039757255593506567270537230,$$
  $$98156413785948877596533722507100341843x + 85481124418552453788443079432675460759)]$$
- $n = 93576491624796965961247428760141142201197042520857824225456053243896853869 82$
- **(H)ECDLP complexity depends on largest prime factor** $r \mid \#\mathrm{Jac}(C)$

Scalar multiplication

- The fundamental operation in curve based public-key cryptography

$$k, P \mapsto [k]P$$

2. Genus 1 vs. Genus 2 (three fights)

Fight #1

NIST's CurveP-256

vs.

Generic1271

NIST's CurveP-256

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$

$b = 41058363725152142129326129780047268409114441015993725554835256314039467401291$

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

$\#E = 115792089210356248762697446949407573529996955224135760342422259061068512044369$

Generic1271

$$p = 2^{127} - 1$$

$p = 170141183460469231731687303715884105727$

$a_3 = 34744234758245218589390329770704207149$, $a_2 = 132713617209345335075125059444256188021$

$a_1 = 90907655901711006083734360528442376758$, $a_0 = 6667986622173728337823560857179992816$

$$C : y^2 = x^5 + a_3x^3 + a_2x^2 + a_1x + a_0$$

$\#\mathrm{Jac} = 28948022309329048848169239995659025138451177973091551374101475732892580332259$

Generic scalar multiplication: double-and-add

- The most simple way to do scalar multiplication is via double-and-add (square-and-multiply for multiplicative notation)

Double-and-add

In: $k = (k_{\ell-1}, \ldots, k_0)_2$, P
Out: $[k]P$

$T \leftarrow P$
for $i = \ell - 2$ **downto** 0 **do**
  $T \leftarrow \textbf{DBL}(T)$
  if $k_i = 1$ **then**
    $T \leftarrow \textbf{ADD}(T, P)$
  end if
end for
return T.

e.g. $k = 18282$

$k =$
$(1,0,0,0,1,1,1,0,1,1,0,1,0,1,0)_2$

so to compute $[k]P$, we ...

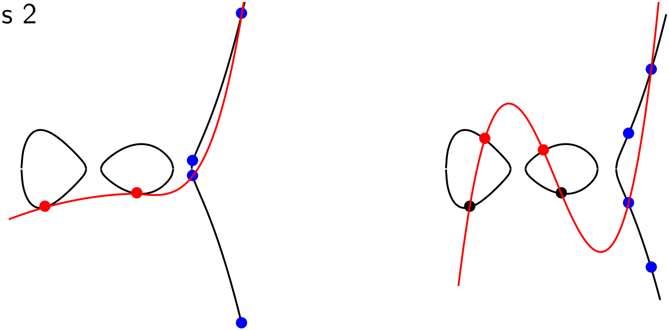
(- , DBL, DBL, DBL,
DBL+ADD, DBL+ADD,
DBL+ADD, DBL,
...,
DBL+ADD,DBL)

- Costs $\lceil \log_2(k) - 1 \rceil$ DBL's and $\approx \frac{1}{2} \log_2(k)$ ADD's

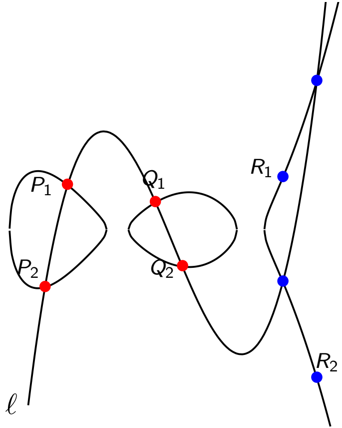
Group operations: elliptic vs. hyperelliptic

Genus 1

Genus 2



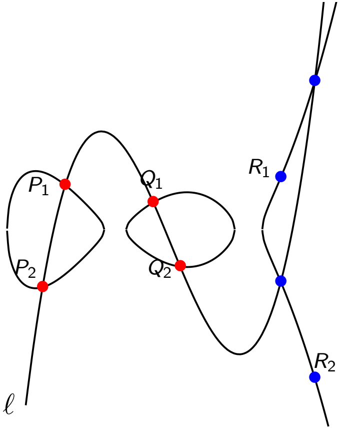
Mumford coordinates



$$sextic = (x - x_{P_1})(x - x_{P_2})(x - x_{Q_1})(x - x_{Q_2})(x - x_{R_1})(x - x_{R_2}) = 0$$
$$\rightarrow quadratic = (x - x_{R_1})(x - x_{R_2}) = 0$$

Computing with actual points means root finding in $\mathbb{F}_q$

Mumford coordinates



$$sextic = (x^2 + \alpha_P x + \beta_P)(x^2 + \alpha_Q x + \beta_Q)(x^2 + \alpha_R x + \beta_R) = 0$$
$$\rightarrow quadratic = (x^2 + \alpha_R x + \beta_R) = 0$$

Mumford coordinates avoid root finding

Results for generic curves

- Formulas for imaginary (degree 5) genus 2 formulas hyperelliptic curves based on C-Lauter'11
- Multiplications (**M**), squarings (**S**) and additions (**a**)

op.	Divisor doubling	Divisor addition	Divisor mix add.
$g = 2$	$34\mathbf{M} + 6\mathbf{S} + 34a$	$44\mathbf{M} + 4\mathbf{S} + 29a$	$37\mathbf{M} + 5\mathbf{S} + 29a$

$\mathbb{F}_p$ operations for common divisor operations in genus 2

- Implementation results (we used windowing - $w = 5$)

implementation	prime p	cycles/scalar mult.
`NIST CurveP-256`	$2^{256} - 2^{224} + \cdots - 1$	658,000
`generic128`	$2^{128} - 173$	364,000
`generic127`	$2^{127} - 1$	248,000

Timings on Intel Core i7-3520M (Ivy Bridge) at 2893.484 MHz

Fight #2

`GLV-j=0`

vs.

BuhlerKoblitzGLV

Gallant-Lambert-Vanstone (GLV) curves

2GLV-j=0 (used by Longa-Sica)

$$p = 2^{256} - 11733$$

p =115792089237316195423570985008687907853269984665640564039457584007913129628203

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

$\#E$ = 115792089237316195423570985008687907852887557187491743187825303095426045639107

Buhler-Koblitz 4GLV curve

$$p = 2^{64} \cdot (2^{63} - 27443) + 1$$

p =170141183460469231731687303715884105727

$$C : y^2 = x^5 + 17$$

$\#\mathrm{Jac}$ =28948022309328876595115567994214488524823328209723866335483563634241778912751

4-GLV: e.g. Buhler-Koblitz curves

- Let $p = 2^{64} \cdot (2^{63} - 27443) + 1$, and let
$$C/\mathbb{F}_p : y^2 = x^5 + 17$$
- $\#\mathrm{Jac} = 28948022309328876595115567994214488524823328209723866335483563634241778912751$
- Notice that $(x, y) \in C \implies (\xi_5 x, y) \in C$, where $\xi_5^5 = 1$,
- It induces a map on $\mathrm{Jac}(C)$ (Mumford coordinates):
$$\phi : (x^2 + u_1 x + u_0, v_1 x + v_0) \mapsto (x^2 + \xi_5 u_1 x + \xi_5^2 u_0, \xi_5^4 v_1 x + v_0)$$
- For $D \in \mathrm{Jac}(C)$, we get the scalar multiples $\phi(D) = [\lambda]D$, $\phi^2(D) = [\lambda^2]D$ and $\phi^3(D) = [\lambda^3]D$ "for free"
- $[k]D$ as $[k]D = [k_0]D + [k_1]\phi(D) + [k_2]\phi^2(D) + [k_3]\phi^3(D)$
- eg. $k = 23477399837278936923599493713286470955314785798347519197199578120259089016680$
$(k_0, k_1, k_2, k_3) =$
$(-6344646642321980551, -3170471730617986668, -4387949940648063094, 3721725683392112311)$
- getting k_i's very quick (CVP in $\mathcal{L} \subset \mathbb{Z}^4$) ...

The GLV lattice

- $r = 28948022309328876595115567994214488524823328209723866335483563634241778912751$
- $\lambda = 78315468676855127052976159806517945867532292413107653204061477837087562856 46$
- GLV lattice $\mathcal{L} \subset \mathbb{Z}^4$ generated by

$$\begin{pmatrix} r & 0 & 0 & 0 \\ -\lambda & 1 & 0 & 0 \\ -\lambda^2 & 0 & 1 & 0 \\ -\lambda^3 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \phi \\ \phi^2 \\ \phi^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \bmod r$$

- Precompute shortest vector $\alpha \in \mathcal{L}$, $\alpha =$

  $(1842396791834961166, 1575206383572171873, -11974991605838508030, 3964086738067825 33)$
- Use α to find vector $(\rho_0, \rho_1, \rho_2, \rho_3) \in \mathcal{L}$ close to $(k, 0, 0, 0) \notin \mathcal{L}$, and take

$$(k_0, k_1, k_2, k_3) = (k, 0, 0, 0) - (\rho_0, \rho_1, \rho_2, \rho_3),$$

  where $||(k_0, k_1, k_2, k_3)||_\infty \leq ||\alpha||_\infty$ in $\mathbb{Z}^4$
- Scalars could be up to $r - 1 = 254$ bits, but $||\alpha||_\infty = 64$ bits

4-GLV: e.g. Buhler-Koblitz curves

- k was 254 bits, but instead we multiexponentiate by

$$\begin{array}{rlr} D & k_0 = [1,0,1,1,0,0,0,0,0,0,0,1,\ldots] & (63\ bits) \\ \phi(D) & k_1 = [0,1,0,1,0,1,1,1,1,1,1,1,\ldots] & (63\ bits) \\ \phi^2(D) & k_2 = [0,1,1,1,1,0,0,1,1,1,0,0,\ldots] & (63\ bits) \\ \phi^3(D) & k_3 = [0,1,1,0,0,1,1,1,0,1,0,0,\ldots] & (63\ bits) \end{array}$$

- **Straus-Shamir multiexponentiation:** $254\texttt{DBL} + 127\texttt{ADD} \rightarrow$
  $\rightarrow 63\texttt{DBL} + 80\texttt{ADD}$

implementation	prime p	cycles/scalar mult.
2GLV-LongaSica	$2^{256} - 11733$	145,000
4GLV-BK	$2^{128} - 24935$	164,000
4GLV-BK	$2^{64} \cdot (2^{63} - 27443) + 1$	156,000

Timings on Intel Core i7-3520M (Ivy Bridge) at 2893.484 MHz

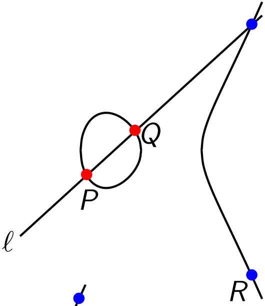
Fight #3

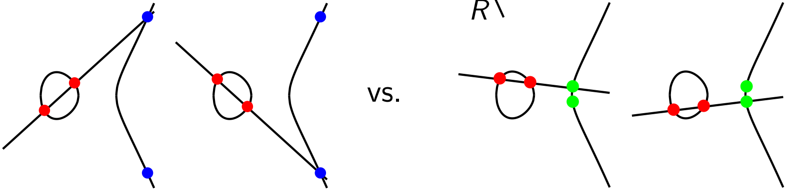
`curve25519`

vs.

Kummer1271

- Can compute $P + Q$ from $\{P, Q, P - Q\}$ without y-coords
- **Key:** to compute $[k]P$, have $[n+1]P$ and $[n]P$ at each stage





same difference → same result

different difference → different result

Genus 2 analogue: the Kummer surface $\mathcal{K}$

- Montgomery identified $P = (P_x, P_y)$ and $-P = (P_x, -P_y)$
- Smart-Siksek'99: $g = 2$ analogue... $\mathrm{Jac}(C) \to \mathcal{K}$ is 2-to-1
- Embedding of $\mathrm{Jac}(C)$ usually into $\mathbb{P}^{15}$

  Flynn: **72 quadratic forms in 16 variables!!!!**
- BUT, $\mathrm{Jac}(C)/\{-\}$ embeds into $\mathbb{P}^3$

  1 equation in 4 variables!!!!
- Gaudry'07: much faster Kummer surface from classical Riemann theta function

"The" Kummer surface $\mathcal{K}$ (Cosset'10)

$$Exyzt = ((x^2 + y^2 + z^2 + t^2) - F(xt + yz) - G(xz + yt) - H(xy + zt))^2$$

- E, F, G, H - functions of $\vartheta_1(0)^2, \vartheta_2(0)^2, \vartheta_3(0)^2, \vartheta_4(0)^2$
- projective point $(x\colon y\colon z\colon t) = (\vartheta_1(\mathbf{z})^2, \vartheta_2(\mathbf{z})^2, \vartheta_3(\mathbf{z})^2, \vartheta_4(\mathbf{z})^2)$

Fast "pseudo-group" operations on $\mathcal{K}$

doubling on $\mathcal{K}$

$(X : Y : Z : T) = [2](x : y : z : t)$

$$
\begin{aligned}
x' &= (x+y+z+t)^2 \\
y' &= (x+y-z-t)^2 \cdot c_y \\
z' &= (x-y+z-t)^2 \cdot c_z \\
t' &= (x-y-z+t)^2 \cdot c_t \\
X &= (x'+y'+z'+t') \\
Y &= (x'+y'-z'-t') \cdot c'_y \\
Z &= (x'-y'+z'-t') \cdot c'_z \\
T &= (x'-y'-z'+t') \cdot c'_t
\end{aligned}
$$

differential addition on $\mathcal{K}$

$(X : Y : Z : T) = (x : y : z : t) + (\underline{x} : \underline{y} : \underline{z} : \underline{t})$
with difference $(\overline{x} : \overline{y} : \overline{z} : \overline{t})$

$$
\begin{aligned}
x' &= (x+y+z+t) \cdot (\underline{x}+\underline{y}+\underline{z}+\underline{t}) \\
y' &= (x+y-z-t) \cdot (\underline{x}+\underline{y}-\underline{z}-\underline{t}) \\
z' &= (x-y+z-t) \cdot (\underline{x}-\underline{y}+\underline{z}-\underline{t}) \\
t' &= (x-y-z+t) \cdot (\underline{x}-\underline{y}-\underline{z}+\underline{t}) \\
X &= (x'+y'+z'+t')^2/\overline{x} \\
Y &= (x'+y'-z'-t')^2/\overline{y} \\
Z &= (x'-y'+z'-t')^2/\overline{z} \\
T &= (x'-y'-z'+t')^2/\overline{t}
\end{aligned}
$$

- Come from Riemann relations (hence "beautiful symmetry")
- No longer a group, but enough to do secure crypto (e.g. DH)
- Each ladder step needs $\texttt{DBL}_{\mathcal{K}} +$ "$\texttt{ADD}$"$_{\mathcal{K}}$ – **only 25 $\mathbb{F}_p$ muls !!!**
- Compare to Mumford – $\texttt{DBL} \approx 40$ and $\texttt{ADD} \approx 50$

Laddering curves

Bernstein's curve25519

$$p = 2^{255} - 19$$

$$p = 57896044618658097711785492504343953926634992332820282019728792003956564819949$$

$$E : y^2 = x^3 + 486662x^2 + x$$

$$\#E = 2^3 \cdot 7237005577332262213973186563042994240857116359379907606001950938285454250989$$

$$\#E' = 2^2 \cdot 14474011154664524427946373126085988481603263447650325797860494125407373907997$$

Kummer1271 (Gaudry-Schost'12)

$$p = 2^{127} - 1$$

$$p = 170141183460469231731687303715884105727$$

$$E = 37299146226279590906389874065895056737, \quad F = 145242473685766417331928186098925456110$$

$$G = 81667768061025231231209905783624370749, \quad H = 54058235547640725801037772083642107170$$

$$Exyzt = ((x^2+y^2+z^2+t^2) - F(xt+yz) - G(xz+yt) - H(xy+zt))^2$$

$$\#\mathrm{Jac}(C) = 2^4 \cdot 1809251394333065553571917326471206521441306174399683558571672623546356726339$$

$$\#\mathrm{Jac}(C') = 2^4 \cdot 1809251394333065553414675955050290598923508843635941313077767297801179626051$$

Performance of Kummer1271

implementation	prime p	cycles/scalar mult.
`curve25519`	$2^{255} - 19$	182,000
`Kummer1271`	$2^{127} - 1$	117,000

Timings on Intel Core i7-3520M (Ivy Bridge) at 2893.484 MHz

- Kummer1271 fastest implementation (in genus 1 or 2) over prime field targeting 128-bit security level

Twist-security

- Recall from two slides ago ...
  - `curve25519` had $\#E = 2^3 \cdot r$ and $\#E' = 2^2 \cdot r'$
  - `kummer1271` had $\#\mathrm{Jac}(C) = 2^4 \cdot r$ and $\#\mathrm{Jac}(C') = 2^4 \cdot r'$
- Why do we need the twist to have *strong* order too?
- **curve25519:** for x-coordinate only (i.e. without y), how do we know/check that we're on $E : y^2 = x^3 + Ax^2 + x$?
- Here we have $[k]x = f(x, k, A)$
- Choose any quadratic non-residue γ, then $E' : \gamma y^2 = x^3 + Ax^2 + x$ is ($\cong$ to) "the" quadratic twist E'
- BUT $f(x, \cdot, A)$ works same for E' too! Could attack ECDLP on E' by sending x s.t. $(x, \pm y) \in E'$
- Same for Kummer in genus 2- could choose $(x : y : z : t) \in \mathcal{K}$ such that pullback goes to $\mathrm{Jac}(C')$, not $\mathrm{Jac}(C)$
- BUT ... safe if curve and twist have good group orders

Summary: genus 1 vs. genus 2 over prime fields

Performance Summary

g	implementation	prime p	cycles	CT	protocols
1	CurveP-256	$2^{256} - 2^{224} + \cdots - 1$	658,000	$\times$	all
	2GLV	$2^{256} - 11733$	145,000	$\times$	all
	curve25519	$2^{255} - 19$	182,000	$\checkmark$	some
2	generic1271	$2^{127} - 1$	248,000	$\times$	all
	4GLV-BK	$2^{64} \cdot (2^{63} - 27443) + 1$	156,000	$\times$	all
	Kummer1271	$2^{127} - 1$	117,000	$\checkmark$	some

Timings on Intel Core i7-3520M (Ivy Bridge) at 2893.484 MHz

- See eBACS for more numbers: `http://bench.cr.yp.to`
- **CT** = "constant time" - resistant to simple power analysis (SPA) attacks, i.e. input independent
- laddering algorithms can't perform additions, so only suitable for some protocols (e.g. DH, ElGamal, but not signatures)

Informal Summary

For all the hard work that it takes to understand/**find!!!**/implement genus 2 cryptography, there are ample rewards, e.g.:

- larger endomorphism ring (4-GLV possible in genus 2, only 2-GLV in genus 1)
- relative benefit from the Kummer surface (laddering) much greater in genus 2
- over prime fields, $g = 2$ gets the Mersenne prime $p = 2^{127} - 1$
- above timings were for 64-bit platforms only... over 32-bit/8-bit architectures, genus 2 would perform even better

BUT ... genus 2 still has its (comparative) drawbacks as well ...

3. Three worthwhile problems in genus 2

Open question #1 - GLV on the Kummer

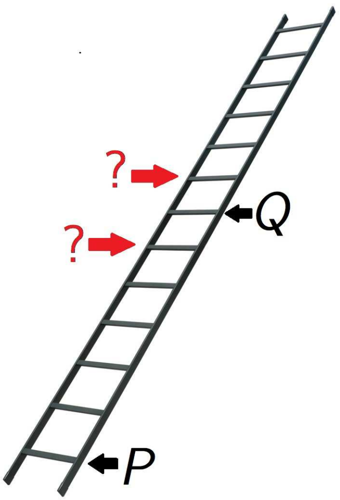
- Using endomorphisms gives big speedups: 364,000 → 156,000
- Using Kummer surface gives big speedups: 248,000 → 117,000
- **Question: can we use endomorphisms on the Kummer surface?**
- Gaudry also noticed that certain Kummers can have an endomorphism ϕ... recall the formulas for Kummer doubling

$$
\begin{aligned}
x' &= (x+y+z+t)^2 \\
y' &= (x+y-z-t)^2 \cdot c_y \\
z' &= (x-y+z-t)^2 \cdot c_z \\
t' &= (x-y-z+t)^2 \cdot c_t
\end{aligned}
\qquad
\begin{aligned}
X &= (x'+y'+z'+t') \\
Y &= (x'+y'-z'-t') \cdot c_y' \\
Z &= (x'-y'+z'-t') \cdot c_z' \\
T &= (x'-y'-z'+t') \cdot c_t'
\end{aligned}
$$

- If $c_y = c_y'$, $c_z = c_z'$, $c_t = c_t'$, then $[2] = \phi \circ \phi$ on $\mathcal{K}$, so $\phi = [\sqrt{2}]$ on $\mathcal{K}$
- Computing $\phi(P) = [\sqrt{2}]P$ on $\mathcal{K}$ is very fast, so can we now do GLV?

Open question #1 - cont

- **Problem:** since we can't add, we can't combine P and Q to emulate multiexponentiation
- We need $Q - P$ or $Q + P$ (quickly!) to kickstart *differential addition chain*
- i.e. We need efficient way of computing $\phi - 1$ or $\phi + 1$ on $\mathcal{K}$



Open question #2 - true resistance

- Suppose genus 2 curves were to be deployed tomorrow
- One serious drawback/problem is how to make genus 2 code *truly side-channel resistant*
- Cantor's algorithm works for any input, but is very "branchy" – simple timing or power attacks can be used
- Implementing full-degree formulas (for weight 2 divisors) is enough for all honest parties – will never run into special cases (prob $\approx 1/p$)
- **BUT**: attackers can recover secret keys quite easily by making us run into special cases

Open question #2 - true resistance

- Suppose Bob's secret key is $k = (k_{\ell-1}, \ldots, k_0)_2$
- Alice chooses a degenerate divisor $D = (x - x_P, y_P)$, computes and sends Bob $\tilde{D} = [\frac{1}{3}]D = (x^2 + \alpha x + \beta, \gamma x + \nu)$.
- **if** *something goes wrong* **then**

  $k_{\ell-2} = 1$

  else

  $k_{\ell-2} = 0$

- w.l.o.g. $k_{\ell-2} = 1$, then Alice now sends $D' = (x - x_{P'}, y_{P'})$, computes and sends $\tilde{D}' = [\frac{1}{7}]D' = (x^2 + \alpha' x + \beta', \gamma' x + \nu')$.
- **Alice can easily reconstruct the key if Bob's code doesn't handle degenerate divisors properly (or in constant time)!!!**

Open question #2 - cont.

- For genus 2 to be a viable off-the-shelf alternative (or preference) . . . **we really need code that . . .**

  1. covers (or at the very least can detect) all cases
  2. runs in constant time / constant power / input independent
  3. is still fast ☺

- Kummer surface code seems to (or does it?)

- But what about the more versatile, more general implementations?

- Whether this solution comes mathematically/programatically/pragmatically, it would most certainly be welcome for genus 2 crypto.

Open question #3 - cont.

One thing that elliptic curves have that genus 2 doesn't is a plethora of non-Weierstrass models, e.g:

- Edwards: $x^2 + y^2 = 1 + dx^2y^2$
- Hessian: $x^3 + y^3 + 1 = dxy$
- Jacobi-quartic: $y^2 = dx^4 + ax^2 + 1$
- ...etc etc ...

Question:

Are there alternative models of genus 2 curves/Jacobians that offer faster arithmetic than $\mathrm{Jac}(C)$ of $C : y^2 = x^5 + \cdots + a_1x + a_0$ in standard Mumford coordinates?

THANKS!!!

see Bos-C-Hisil-Lauter: "Fast Cryptography in Genus 2"

http://eprint.iacr.org/2012/670