Advanced Constructions in Curve-based Cryptography

Benjamin Smith

Team **GRACE**

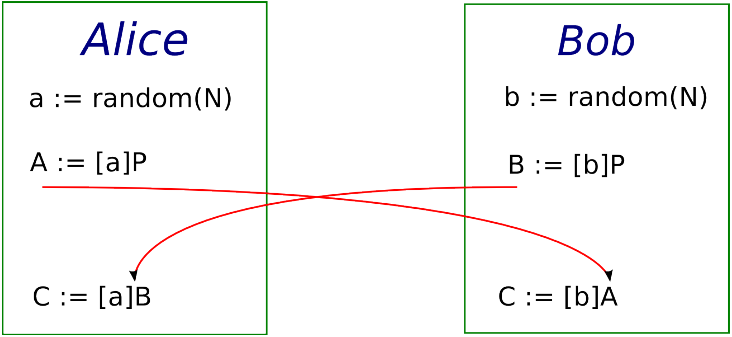
INRIA and **Laboratoire d'Informatique de l'École polytechnique (LIX)**

Summer school on real-world crypto and privacy

Sibenik, Croatia, June 9 2016

Diffie–Hellman key exchange: classic view

$\mathcal{G} = \langle P \rangle$ is a cyclic group; a, b secret integers.



Security: Computational Diffie–Hellman Problem (CDHP)

Given $(P, [a]P, [b]P)$, find $[ab]P$.

Practical cryptographic groups $\mathcal{G}$: CDHP $\equiv$ Discrete Log

Uniformity

All the hard work is in the scalar multiplication:
we need to make this fast.

For key generation and signing, pure speed is not enough:
we need to avoid (at least) basic side-channel attacks.
This means we need constant-time implementations.

On an algorithmic level, we need *uniformity*:
the number and order of instructions must be
exactly the same for every input.

We can assume that all scalars have the same bitlength
(padding top bits with 0, adding multiples of N, etc.)

Start dumb, get smarter.

Algorithm 1 Classic double-and-add scalar multiplication

function $\textsc{Naive}(m = \sum_{i=0}^{\beta-1} m_i 2^i, P)$
  $R \leftarrow \mathcal{O}_{\mathcal{E}}$
  for $i := \beta - 1$ down to 0 **do**
    $R \leftarrow [2]R$
    if $m_i = 1$ **then**
      $R \leftarrow R \oplus P$
    end if
  end for $\triangleright$ invariant: $R = ([\lfloor m/2^i \rfloor])P$
  return R $\triangleright$ $R = [m]P$
end function

Problem: we only add when $m_i = 1$, revealing secret bits.

The Montgomery ladder

Montgomery's simple differential addition chain:

Algorithm 2 The Montgomery ladder

function $\textsc{Ladder}(m = \sum_{i=0}^{\beta-1} m_i 2^i,\ P)$
    $(R_0, R_1) \leftarrow (\mathcal{O}_{\mathcal{E}}, P)$
    for $i := \beta - 1$ down to 0 **do**
        if $m_i = 0$ **then**
            $(R_0, R_1) \leftarrow ([2]R_0, R_0 \oplus R_1)$
        else $\triangleright$ $m_i = 1$
            $(R_1, R_0) \leftarrow ([2]R_1, R_0 \oplus R_1)$
        end if
    end for $\triangleright$ invariant: $(R_0, R_1) = ([\lfloor m/2^i \rfloor]P, [\lfloor m/2^i \rfloor + 1]P)$
    return R_0 $\triangleright$ $R_0 = [m]P,\ R_1 = [m]P \oplus P$
end function

Safety

We now have a uniform sequence of doubles and adds.

To make this a truly uniform/constant-time algorithm, we convert the **if** statement into conditional swaps (see Peter Schwabe's talk on Friday).

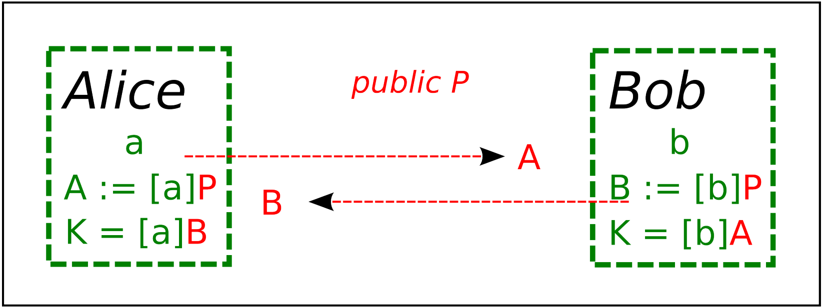
We also need uniform $[2]$ and $\oplus$.

No exceptional cases, no branches, no failures.

$\implies$ Use e.g. the group laws from Monday's talk.

We now have a start towards safe, fast implementations of protocols using elliptic curves...

A modern view of Diffie–Hellman



- $\mathcal{G}$ is just a set, not a group!
- $[a]$, $[b]$ secret commuting maps $\mathcal{G} \to \mathcal{G}$.
- CDHP: reduce to CDHP/Discrete Log in groups.

Candidates for Diffie–Hellman systems

1970s/80s *Set* $\mathcal{G}$: subgroup of $\mathbb{G}_m(\mathbb{F}_p)$.

Maps $[a]$, $[b]$*:* random exponentiations.

Requires: hard CHDP in $\mathbb{G}_m(\mathbb{F}_p)$.

90s/2000s *Set* $\mathcal{G}$: subgroup of an elliptic curve $\mathcal{E}(\mathbb{F}_p)$

Maps $[a]$, $[b]$*:* random scalar multiplications.

Requires: hard CDHP in $\mathcal{E}(\mathbb{F}_p)$.

Advantage: MUCH smaller $q \implies$ faster, compact.

2006$\rightarrow$ *Set* $\mathcal{G} = \mathbb{P}^1(\mathbb{F}_p)$ ($=$ ($\log_2 q$)*-bit strings*) $= (\mathcal{E}/\langle \pm 1 \rangle)(\mathbb{F}_p)$.

Maps $[a]$, $[b]$*:* random commuting $\mathbb{P}^1 \to \mathbb{P}^1$ (from $\mathcal{E}$).

Requires: hard CDHP in $\mathcal{E}(\mathbb{F}_p)$ and $\mathcal{E}'(\mathbb{F}_p)$ (quad. twist)

Advantage: much faster, more compact, fault-tolerant.

Moving from $\mathcal{E}$ to $\mathbb{P}^1 = \mathcal{E}/\langle\pm 1\rangle$

Quotient map $x : \mathcal{E} \longrightarrow \mathbb{P}^1 = \mathcal{E}/\langle\pm 1\rangle$.

The group law $\oplus$ on $\mathcal{E}$ is lost on $\mathbb{P}^1$...
...but for any $m \in \mathbb{Z}$
we have a well-defined "scalar multiplication"

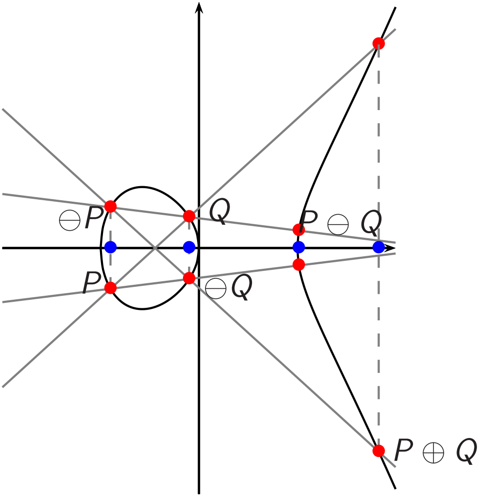
$$[m]_* : x(P) \longmapsto x([m]P) ,$$

because $\ominus[m](P) = [m](\ominus P)$.

Problem: How do we compute $[m]_*$ efficiently, *without* $\oplus$?

Observe:
$\{x(P), x(Q)\}$ determines $\{x(P \oplus Q), x(P \ominus Q)\}$.

$\{x(P), x(Q)\}$ *determines* $\{x(P \ominus Q), x(P \oplus Q)\}$



...and any 3 of $x(P)$, $x(Q)$, $x(P \ominus Q)$, $x(P \oplus Q)$ determines the 4th

Since any 3 of $x(P)$, $x(Q)$, $x(P \ominus Q)$, $x(P \oplus Q)$ determines the 4th, we can define

pseudo-addition

$$\mathtt{xADD} : (x(P), x(Q), x(P \ominus Q)) \longmapsto x(P \oplus Q)$$

pseudo-doubling

$$\mathtt{xDBL} : x(P) \longmapsto x([2]P)$$

$\Longrightarrow$ Evaluate $[m]_*$ by combining `xADD`s and `xDBL`s using differential addition chains

(ie. every $\oplus$ has summands with known difference)

(Luckily, we already know one of these...)

The Montgomery ladder, again

Algorithm 3 The Montgomery ladder

function $\textsc{Ladder}(m = \sum_{i=0}^{\beta-1} m_i 2^i,\ P)$
  $(R_0, R_1) \leftarrow (\mathcal{O}_{\mathcal{E}}, P)$
  for $i := \beta - 1$ down to 0 **do**
    if $m_i = 0$ **then**
      $(R_0, R_1) \leftarrow ([2]R_0, R_0 \oplus R_1)$
    else ▷ $m_i = 1$
      $(R_1, R_0) \leftarrow ([2]R_1, R_0 \oplus R_1)$
    end if
  end for ▷ invariant: $(R_0, R_1) = ([\lfloor m/2^i \rfloor]P, [\lfloor m/2^i \rfloor + 1]P)$
  return R_0 ▷ $R_0 = [m]P,\ R_1 = [m]P \oplus P$
end function

For each $R_0 \oplus R_1$, the difference $R_0 \ominus R_1$ is *fixed*.

The x-only Montgomery ladder

Algorithm 4 The Montgomery ladder

function $\textsc{Ladder}(m = \sum_{i=0}^{\beta-1} m_i 2^i,\ P)$
  $(R_0, R_1) \leftarrow (\mathcal{O}_{\mathcal{E}}, x(P))$
  for $i := \beta - 1$ down to 0 **do**
    if $m_i = 0$ **then**
      $(R_0, R_1) \leftarrow (\mathtt{xDBL}(R_0), \mathtt{xADD}(R_0, R_1, x(P))$
    else $\triangleright$ $m_i = 1$
      $(R_1, R_0) \leftarrow (\mathtt{xDBL}(R_1), \mathtt{xADD}(R_0, R_1, x(P))$
    end if
  end for $\triangleright$ invariant: $(R_0, R_1) = (x([\lfloor m/2^i \rfloor]P), x([\lfloor m/2^i \rfloor + 1]P))$
  return R_0 $\triangleright$ $R_0 = [m]_*(x(P)) = x([m]P),\ R_1 = x([m]P \oplus P)$
end function

Note: the xDBL and xADD involve some shared operands.
We usually combine them in a faster xDBLADD operation.

Montgomery models of elliptic curves

$$\mathcal{E} : \Delta Y^2 Z = X(X^2 + (4C - 2)XZ + Z^2)$$

with curve constant C and "twisting constant" Δ in $\mathbb{F}_p$.

The map $x : \mathcal{E} \to \mathbb{P}^1$ is $x : (X : Y : Z) \longmapsto (X : Z)$.

- `xADD`$((X_P : Z_P), (X_Q : Z_Q), (X_{P\ominus Q} : Z_{P\ominus Q}))$
  $= \left(Z_{P\ominus Q}(S_P T_Q + T_P S_Q)^2 : X_{P\ominus Q}(S_P T_Q - T_P S_Q)^2\right)$
  where $S_P := X_P - Z_P$, $T_P := X_P + Z_P$, etc.
- `xDBL`$((X : Z)) = (UV : W(U + CW))$
  where $U = (X + Z)^2$, $V = (X - Z)^2$, $W = U - V$.

Observe that Δ *never appears* in these operations!

Quadratic twists

Consider two curves with the same C:

$$\mathcal{E} : \Delta Y^2 Z = X(X^2 + (4C - 2)XZ + Z^2) ,$$
$$\mathcal{E}' : \Delta' Y^2 Z = X(X^2 + (4C - 2)XZ + Z^2) .$$

Isomorphic via $(X : Y : Z) \longmapsto (X : \sqrt{\Delta/\Delta'} \cdot Y : Z)$

—but if Δ/Δ' is not a square in $\mathbb{F}_p$, then

$\mathcal{E}$ and $\mathcal{E}'$ are only isomorphic over $\mathbb{F}_{p^2}$ and not $\mathbb{F}_p$!

In this case, we say $\mathcal{E}$ and $\mathcal{E}'$ are *quadratic twists*.

Quadratic twists are unique up to $\mathbb{F}_p$-isomorphism (since in $\mathbb{F}_p$, the product of *any* two non-$\square$ is a $\square$); so we generally choose one, and say *The* quadratic twist.

Quadratic twists

$$\mathcal{E} : \Delta Y^2 Z = X(X^2 + (4C - 2)XZ + Z^2)$$
$$\mathcal{E}' : \Delta' Y^2 Z = X(X^2 + (4C - 2)XZ + Z^2)$$

Suppose $(\Delta/\Delta' \neq \square)$; then $\mathcal{E}$ and $\mathcal{E}'$ have the same "geometry", but their groups $\mathcal{E}(\mathbb{F}_p)$ and $\mathcal{E}'(\mathbb{F}_p)$ are generally different.

At infinity: $(1:0) = x(\mathcal{O}_{\mathcal{E}}) = x(\mathcal{O}_{\mathcal{E}})$. For each α in $\mathbb{F}_p$, either:

- $(\alpha : 1) = x(P) = x(\ominus P)$ for some $P \in \mathcal{E}(\mathbb{F}_p)$, or
- $(\alpha : 1) = x(P') = x(\ominus P')$ for some $P' \in \mathcal{E}'(\mathbb{F}_p)$, or
- Both! $\implies Y(P) = Y(P') = 0$, so P and P' have order 2.

This also implies $\#\mathcal{E}(\mathbb{F}_p) + \#\mathcal{E}'(\mathbb{F}_p) = 2(p+1)$.

Since Δ and Δ' *never appear* in `xDBL` or `xADD`,
$\implies$ `xDBL`, `xADD` are identical for $\mathcal{E}/\langle\pm1\rangle$ and $\mathcal{E}'/\langle\pm1\rangle$.

For all $\alpha \in \mathbb{F}_p$, we have $\alpha \in x(\mathcal{E}(\mathbb{F}_p))$ or $\alpha \in x(\mathcal{E}'(\mathbb{F}_p))$.
$\implies$ feeding arbitary input bitstrings to $[a]_*$ and $[b]_*$ amounts to taking $\mathcal{G} = \mathcal{E}(\mathbb{F}_p)/\langle\pm1\rangle \cup \mathcal{E}'(\mathbb{F}_p)/\langle\pm1\rangle$.

Allowing arbitrary inputs is important in defending against fault attacks (where inputs and variables are modified)

Now $\mathcal{E}(\mathbb{F}_p)$ and $\mathcal{E}'(\mathbb{F}_p)$ must *both* have hard CDHP/DLPs
—in this case, we say $\mathcal{E}$ is twist-secure.

This is the basis of Bernstein's `Curve25519` software.

What are the elliptic curves doing?

Diffie–Hellman is now defined by "secret functions" $[a]_*$ and $[b]_*$, each of which is just a series of $\log_2 q$ random CSwaps followed by

$$(T_0, T_1) \longmapsto (\mathtt{xDBL}(T_0), \mathtt{xADD}(T_0, T_1, X)).$$

where $X = x(P)$, A, or B, depending on the protocol step.

One system parameter, $C \in \mathbb{F}_p \longleftrightarrow$ curve-twist pair $(\mathcal{E}, \mathcal{E}')$, which

- Defines the operation xDBL (xADD *is independent of* $\mathcal{E}$, $\mathcal{E}'$)
- Proves that the secret functions $[a]_*, [b]_*$ commute
- Gives hard upper and conjectural lower bounds on security (from the CDHPs on on $\mathcal{E}$ and $\mathcal{E}'$)

Pulling a y-rabbit out of an x-hat

x-only multiplication computes $x([m]P)$ from $x(P)$.

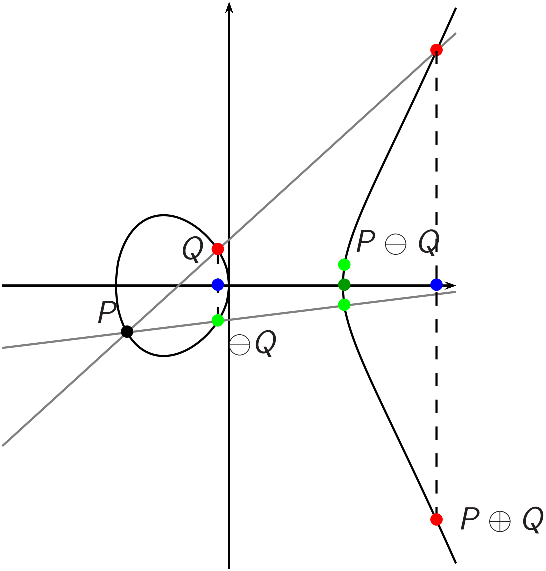
Mathematically, we threw away the sign:
you can't deduce $y([m]P)$ from P and $x([m]P)$.
But if you used the Montgomery ladder, then you can!

At the end of the loop, $R_0 = x([m]P)$ and $R_1 = x([m]P \oplus P)$.
It's not hard to show that P, $x(Q)$, and $x(Q \oplus P)$
uniquely determines $y(Q)$ (for any Q).

Result: the x-only Montgomery ladder can be used for full "signed" scalar multiplication (eg. in signature schemes).

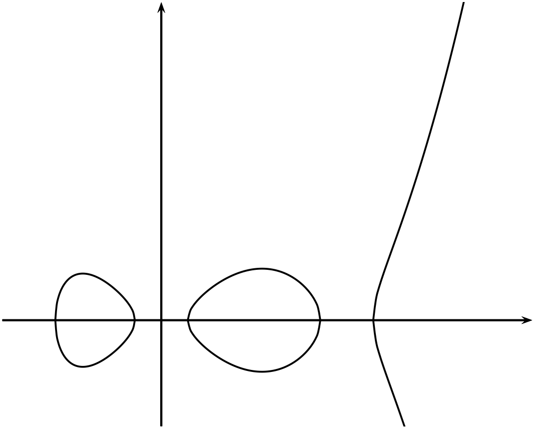
See: Lopez–Dahab, Okeya–Sakurai, Brier–Joye.

P, $x(Q)$, and $x(P \oplus Q)$ determine Q



Genus 2 curves

$\mathcal{X} : y^2 = f(x)$ with $f \in \mathbb{F}_p[x]$ degree 5 or 6 and squarefree



Unlike elliptic curves, the points do not form a group.

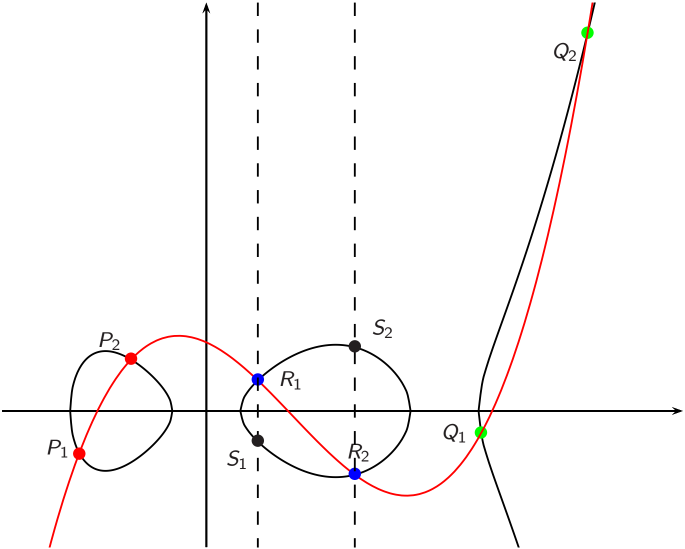
Making groups from genus 2 curves

Jacobian: algebraic group $\mathcal{J}_{\mathcal{X}} \cong \mathrm{Pic}^0(\mathcal{X})$;
geometrically, $\mathcal{J}_{\mathcal{X}} \sim \mathcal{X}^{(2)}$ (symmetric square of $\mathcal{X}$)
(with all pairs $\{(x, y), (x, -y)\}$ "blown down" to 0)

Group law on $\mathcal{J}_{\mathcal{X}}$ induced by
$\{P_1, P_2\} \oplus \{Q_1, Q_2\} \oplus \{R_1, R_2\} = 0$
whenever $P_1, P_2, Q_1, Q_2, R_1, R_2$ are
the intersection of $\mathcal{X}$ with some cubic $y = g(x)$.

Why? 4 points in the plane determine a cubic;
and a cubic $y = g(x)$ intersects $\mathcal{X} : y^2 = f(x)$ in 6 points
because $g(x)^2 = f(x)$ has 6 solutions.

Genus 2 group law: $\{P_1, P_2\} \oplus \{Q_1, Q_2\} = \{S_1, S_2\}$



What is the Jacobian?

$$\mathcal{J}_{\mathcal{X}} \sim \mathcal{X}^{(2)} \implies \mathcal{J}_{\mathcal{X}} \text{ is a surface.}$$

Points in $\mathcal{J}_{\mathcal{X}}(\mathbb{F}_p) \longleftrightarrow$ pairs $\{P_1, P_2\}$ of points of $\mathcal{X}$ with P_1, P_2 both in $\mathcal{X}(\mathbb{F}_p)$ or conjugate in $\mathcal{X}(\mathbb{F}_{p^2})$

$$\implies \#\mathcal{J}_{\mathcal{X}}(\mathbb{F}_p) = O(p^2).$$

More precisely: $(\sqrt{p}-1)^{2\times 2} \le \#\mathcal{J}_{\mathcal{X}}(\mathbb{F}_p) \le (\sqrt{p}+1)^{2\times 2}$.

Replace 2s with 1s $\longrightarrow$ elliptic curves (genus 1).

Abstractly: $\mathcal{J}_{\mathcal{X}}(\mathbb{F}_p)$ drop-in replacement for some $\mathcal{E}(\mathbb{F}_q)$ (but only need $\log_2 p \approx \frac{1}{2}\log_2 q$).

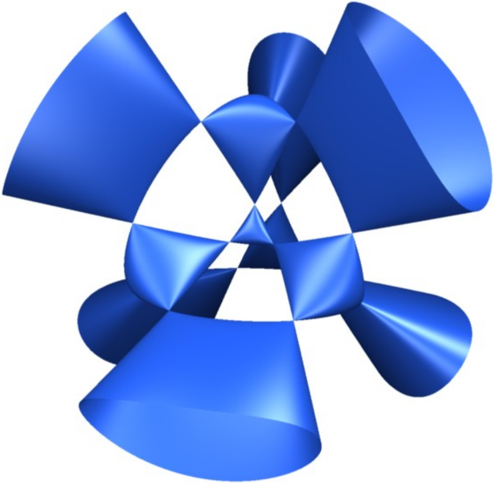
But the algorithms and geometry of $\mathcal{J}_{\mathcal{X}}$ are *much* more complicated than for $\mathcal{E}$.

Kummer varieties

If $\mathcal{E} : y^2 = f(x)$ is an elliptic curve,
then $\ominus(x, y) = (x, -y)$;
so $P \mapsto x(P)$ is the quotient by ± 1.
$\implies$ the x-line $\mathbb{P}^1$ is the *Kummer variety* of $\mathcal{E}$.

Genus 2 analogue of the x-line $\mathbb{P}^1$:
The *Kummer surface* $\mathcal{K}_{\mathcal{X}} := \mathcal{J}_{\mathcal{X}} / \langle \pm 1 \rangle$
is a quartic surface in $\mathbb{P}^3$ with 16 point singularities
(which are the images of the 16 points in $\mathcal{J}_{\mathcal{X}}[2]$*).*

What a Kummer surface looks like



...This is the genus 2 analogue of what is just a line for elliptic curves, which says a lot about the jump in mathematical complexity...

Kummer surfaces

The classical model of the Kummer surface for $\mathcal{X}$:

$$X^4 + Y^4 + Z^4 + W^4 + 2E \cdot XYZW$$
$$= F(X^2W^2 + Y^2Z^2) + G(X^2Z^2 + Y^2W^2) + H(X^2Y^2 + Z^2W^2)$$

where E, F, G, H are algebraic expressions in the coefficients of $\mathcal{X}$.

$\mathcal{K}_{\mathcal{X}}$ is not a group, but we get scalar multiplication from $\mathcal{J}_{\mathcal{X}}$
(since $[m](\pm D) = \pm([m]D)$).

Faster than elliptic x-line arithmetic at the same security level
(Chudnovsky & Chudnovsky, Gaudry)

Eg. 128-bit security: $\mathcal{K}_{\mathcal{X}}$ over 128-bit field
beats $\mathcal{E}$ over 256-bit field

Kummer surface arithmetic

Let 0_K be the image of $0_{\mathcal{J_X}}$ in $\mathcal{K_X}$, and define

$$\mathcal{M} : ((x_1 : y_1 : z_1 : t_1), (x_2 : y_2 : z_2 : t_2)) \longmapsto (x_1 x_2 : y_1 y_2 : z_1 z_2 : t_1 t_2) \ ,$$

$$\mathcal{S} : (x : y : z : t) \longmapsto (x^2 : y^2 : z^2 : t^2) \ ,$$

$$\mathcal{I} : (x : y : z : t) \longmapsto (1/x : 1/y : 1/z : 1/t)$$

and the *Hadamard transformation*

$$\mathcal{H} : (x : y : z : t) \longmapsto (x' : y' : z' : t') \quad \text{where} \quad \begin{cases} x' = x + y + z + t \ , \\ y' = x + y - z - t \ , \\ z' = x - y + z - t \ , \\ t' = x - y - z + t \ . \end{cases}$$

Then we can use the following operations for the Montgomery ladder:

- $\mathtt{xADD}(\pm P, \pm Q, \pm(P \ominus Q))$
  $= \mathcal{M}(\mathcal{HM}(\mathcal{M}(\mathcal{HS}(\pm P), \mathcal{HS}(\pm Q)), \mathcal{IH}(0_K)), \mathcal{I}(\pm(P \ominus Q)))$
- $\mathtt{xDBL}(\pm P) = \mathcal{M}(\mathcal{HM}(\mathcal{S}(\mathcal{HS}(\pm P)), \mathcal{IH}(0_K)), \mathcal{I}(0_K))$

(The green things here are essentially constants)

Kummer surfaces: Theory into practice

Kummer surfaces are already used for high-speed Diffie–Hellman

E.g.: Bos–Costello–Hisil–Lauter, 2012;
Bernstein–Chuengsatiansup–Lange–Schwabe, 2014

μKummer (Renes–Schwabe–S.–Batina, CHES 2016):
Open Kummer surface crypto for 8- and 32-bit microcontrollers.
Efficient Diffie–Hellman *and* Schnorr signatures
(using genus-2 y-recovery analogue, Chung–Costello–S.).

Comparison for 8-bit architecture (AVR ATmega):

Protocol	Object	kCycles	Stack bytes
Diffie–Hellman	Curve25519	3590	548
	μ**Kummer**	2634 (73%)	248 (45%)
Schnorr sign	Ed25519	19048	1473
	μ**Kummer**	10404 (55%)	926 (63%)
Schnorr verif.	Ed25519	30777	1226
	μ**Kummer**	16241 (53%)	992 (75%)

(vs. Curve25519: Düll-Haase-Hinterwälder-Hutter-Paar-Sánchez-Schwabe, Ed25519: Nascimento-López-Dahab)