# Smaller and faster public-key crypto for IoT from genus-2 curves

Benjamin Smith

Real-world crypto+privacy summer school, Sibenik, 18/6/2019

Inria + Laboratoire d'Informatique de l'École polytechnique (LIX)

# Ancient history

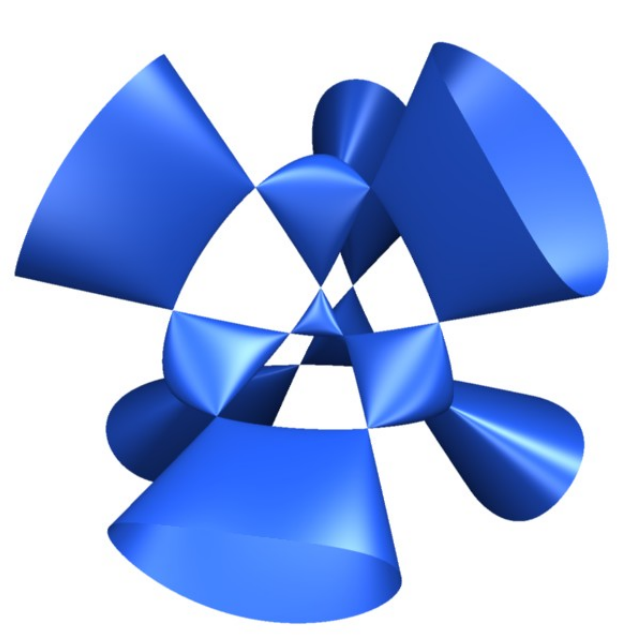
1860s: **Kummer** studies quartic surfaces in $\mathbb{P}^3$ with **16 point singularities** (the maximum possible number).

At first, these surfaces were very useful in **optics**; then they became important examples in **algebraic geometry**.

**Decades later**, a connection was made with **abelian varieties** and Jacobians of genus-2 curves.

**120 years later**, Kummer varieties appeared in cryptography.

# Kummer surfaces



# Elliptic curves

Elliptic curves: $\mathcal{E} : y^2 = x^3 + ax + b$. The points form a group.

**Scalar multiplication** $P \mapsto [m]P = \underbrace{P + \cdots + P}_{m \text{ times}}$.

**Negation** automorphism $-1 : (x, y) \mapsto (x, -y)$.

Take the **quotient** by $\pm 1$, identifying $P$ and $-P$:

$$P \longmapsto \{P, -P\} = \{(x_P, y_P), (x_P, -y_P)\} \leftrightarrow x_P \,.$$

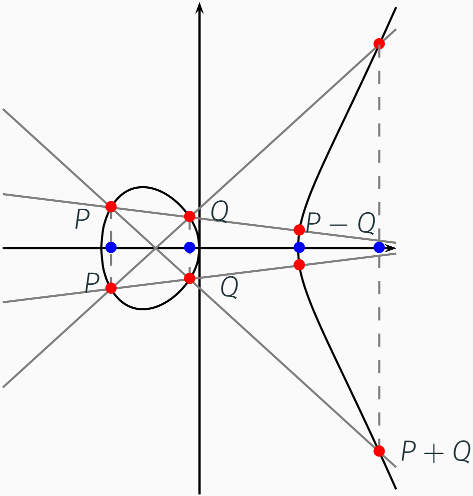
The image of this map is $\mathcal{E}/\langle \pm 1 \rangle \cong \mathbb{P}^1$, the $x$-axis.

Removing 1 bit of “sign” information takes us from $\mathcal{E}$ to $\mathbb{P}^1$, a **much simpler** geometrical object.

On the other hand, it makes using the group law tricky.

# What remains of the group law

$\pm P$ and $\pm Q$ only determine the *pair* $\{\pm(P+Q), \pm(P-Q)\}$.




...and any 3 of $x(P)$, $x(Q)$, $x(P-Q)$, $x(P+Q)$ determines the 4th.

# $x$-only arithmetic

Since any 3 of $x(P)$, $x(Q)$, $x(P-Q)$, $x(P+Q)$ determines the 4th, we can define

**Pseudo-addition**, or xADD:

$$\texttt{xADD} : (x(P), x(Q), x(P-Q)) \longmapsto x(P+Q)$$

**Pseudo-doubling**, or xDBL:

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

To compute the scalar multiple $x([m]P)$ from $m$ and $x(P)$: combine xADDs and xDBLs using the **Montgomery ladder**.

# Genus-2 Jacobians

Genus-2 curves: $\mathcal{C} : y^2 = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$.

The **Jacobian** $\mathcal{J}_\mathcal{C}$ is a group built from $\mathcal{C}$. An **algebraic surface**: almost all elements of $\mathcal{J}_\mathcal{C}$ look like **pairs of points** on $\mathcal{C}$.

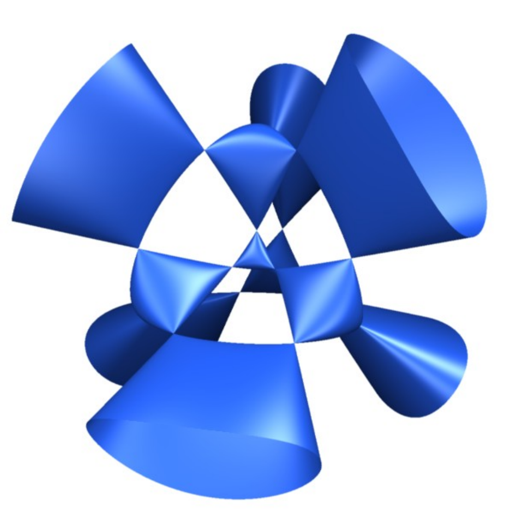
**Negation** acts on *both* $y$-coordinates in a pair:

$$-1 : P = \{(x_1, y_1), (x_2, y_2)\} \longmapsto -P = \{(x_1, -y_1), (x_2, -y_2)\}.$$

Quotient by $\pm 1$ involves symmetric functions $x_1 + x_2$, $x_1x_2$, $y_1y_2$.

The map $P \mapsto (1 : x_1 + x_2 : x_1x_2)$ *would* take us from $\mathcal{J}_\mathcal{C}$ to $\mathbb{P}^2$…
But $y_1y_2$ complicates things.

The quotient object $\mathcal{J}_\mathcal{C}/\langle\pm 1\rangle$ is not $\mathbb{P}^2$, but a **Kummer surface**.



The 16 singularities are the images of the 2-torsion points of $\mathcal{J}_{\mathcal{C}}$ (which are obviously fixed by $\pm 1$).

# Kummer surfaces again

*Kummer surface* $\mathcal{K}_\mathcal{C} := \mathcal{J}_\mathcal{C}/\pm$.

Classical **defining equation**:

$$4E \cdot X_1X_2X_3X_4 = \left( \begin{array}{c} X_1^2 + X_2^2 + X_3^2 + X_4^2 - F \cdot (X_1X_4 + X_2X_3) \\ - G \cdot (X_1X_3 + X_2X_4) - H \cdot (X_1X_2 + X_3X_4) \end{array} \right)^2$$

**Operations** (*green = constant*):

- xADD$(\pm P, \pm Q, \pm(P-Q))$
  $= \mathcal{M}(\mathcal{S}(\mathcal{H}(\mathcal{M}(\mathcal{M}(\mathcal{H}(\pm P), \mathcal{H}(\pm Q)), c))), \mathcal{I}(\pm(P-Q)))$
- xDBL$(\pm P) = \mathcal{M}(\mathcal{S}(\mathcal{H}(\mathcal{M}(\mathcal{S}(\mathcal{H}(\pm P)), c))), c')$

where $\mathcal{M}, \mathcal{S}, \mathcal{I}$ are 4-way parallel multiplies, squares, inversions and

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

# An important question

Given the jump in mathematical complexity, we have to ask:

**Why bother** with genus 2 and Kummer surfaces?

To answer this, let's go back through the history of ECC...

# Elliptic curve cryptography: an approximate history

# The beginning

Big bang: Schoof (1983). A polynomial-time point counting algorithm for elliptic curves.

*The first really modern algorithm for elliptic curves.*

Not used in crypto at the time (ECC hadn't been invented yet!), but the successor of this algorithm (SEA) is vital for generating secure elliptic curves.

# 1985, a very busy year

**1985**: **Hendrik W. Lenstra** announces **ECM** factorization.

Requires a lot of **scalar multiplications** on various $\mathcal{E}$:

1. Compute $P = (X : Y : Z) \longmapsto (X_m : Y_m : Z_m) = [m]P$ for a big smooth $m \in \mathbb{Z}_{>0}$ and $P \in \mathcal{E}(\mathbb{Z}/N\mathbb{Z})$.
2. Finally, compute $\gcd(Z_m, N)$. If this doesn't factor $N$, then take another $\mathcal{E}$ and do more scalar multiplication.

ECM was the first modern elliptic-curve algorithm where general **scalar multiplication** is the **core operation**.

**Key idea**: replace the multiplicative group $\mathbb{F}_p^\times$ in the classic $p-1$ factoring algorithm with an elliptic group $\mathcal{E}(\mathbb{F}_p)$.

# History: The dawn of ECC

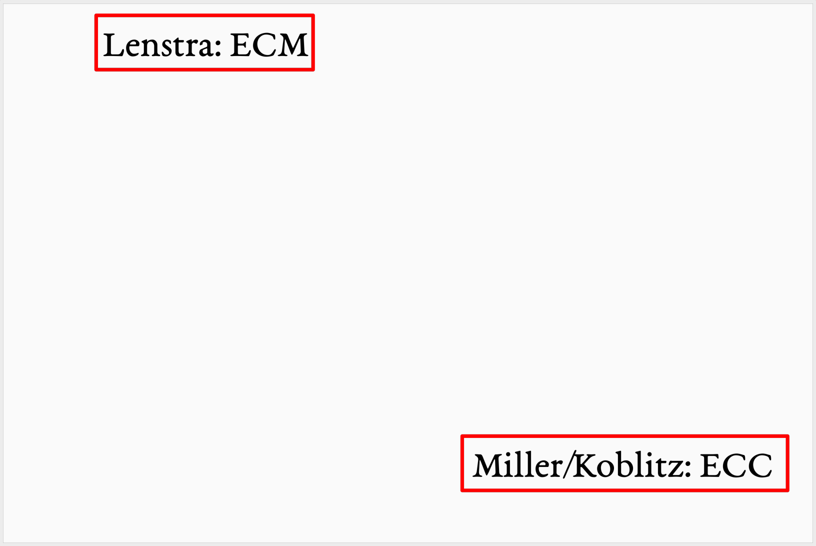
Having seen Schoof and Lenstra's results, by the end of 1985, **Victor Miller** and **Neal Koblitz** had independently set out elliptic curve Diffie–Hellman key exchange (**ECDH**).

In the last paragraph of his CRYPTO 1985 paper, Miller says

> *Finally, it should be remarked, that even though*
> *we have phrased everything in terms of points*
> *on an elliptic curve, that, for the key exchange protocol*
> *(and other uses as one-way functions),*
> *that* ***only the x-coordinate needs to be transmitted****...*
> *the x-coordinate of a multiple depends only on the x-coordinate of the original point.*

Somehow, **cryptographers ignored this**.

# History: Invasion of the Number Theorists



# Montgomery and the Chudnovskys

**By late 1985**: practical improvements to Lenstra's ECM.

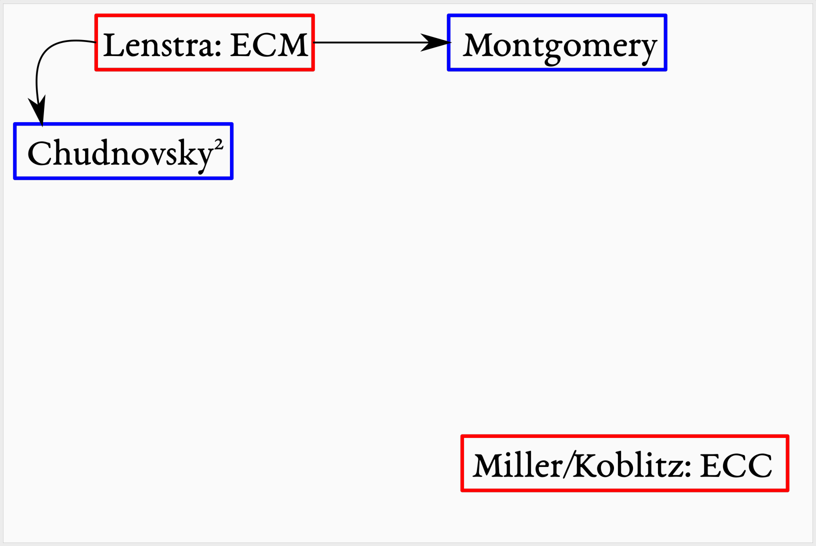
**Peter L. Montgomery**

- Suggested using only $x$-coordinate arithmetic
- Defined new curve form $\mathcal{E} : BY^2 = X(X^2 + AXZ + Z^2)$, **specially tuned** for efficient $x$-only arithmetic

**D. V.** and **G. V. Chudnovsky**

- Compared many classical models of elliptic curves (some with the $x$-coordinate trick)
- Also proposed using more general abelian varieties, showing **Kummer surface** operations as an example

# History: Late 1985



# History: HECC

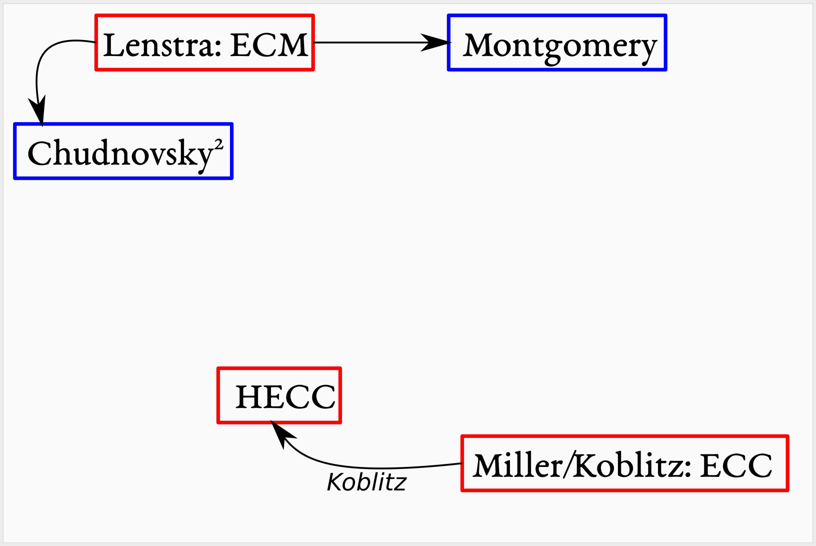
By **1989**: Koblitz suggested using Jacobians of hyperelliptic curves in place of elliptic curves for crypto.

Curve of genus $g$ over $\mathbb{F}_p \implies$ Jacobian with $\sim p^g$ elements.

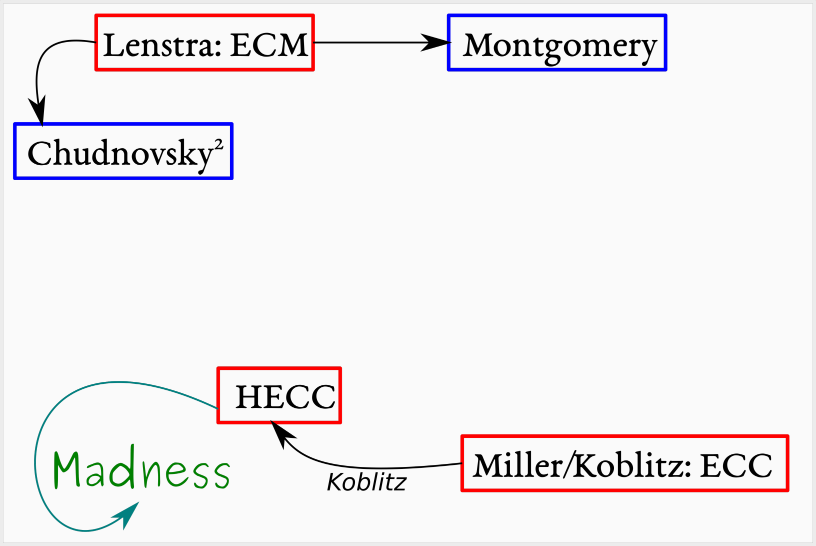
Direct **trade-off** between Jacobian dimension $g$ and field size: smaller fields are much faster to work with.

Later, index calculus algorithms for discrete logs make this a **bad trade for genus** $> 2$.

# History: Party like it's 1989



# History: Things get out of hand



# History: Things get out of hand

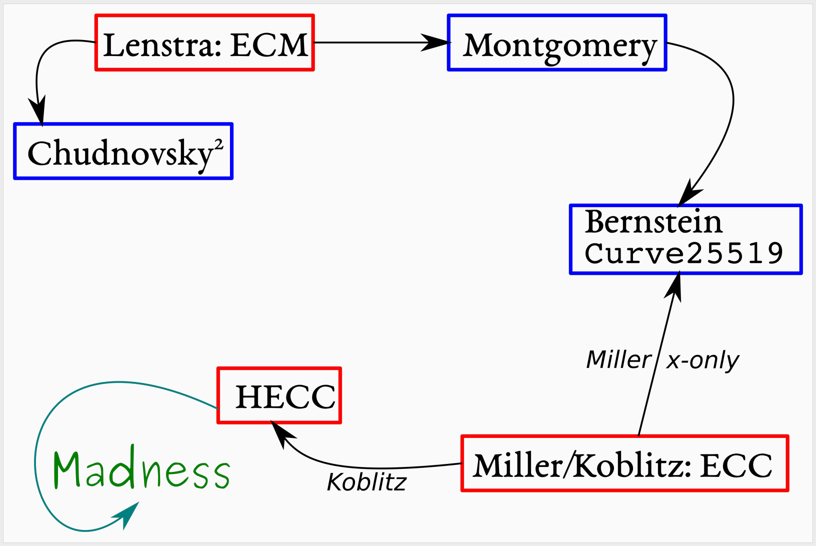
By the mid-2000s, 20 years after Miller and Koblitz:

- Constructive curve-based crypto: ECC and genus-2 HECC.
- Elliptic curves were standardized and started to become really useful.
- But genus-2 crypto parameters weren't quite there yet: point-counting algorithms were still being developed. *(This is still an important research problem!)*

**2005**: Dan Bernstein develops `Curve25519`, which combines

- Miller's $x$-only ECDH idea
- Montgomery's $x$-only ECM algorithms

# History: The need for speed



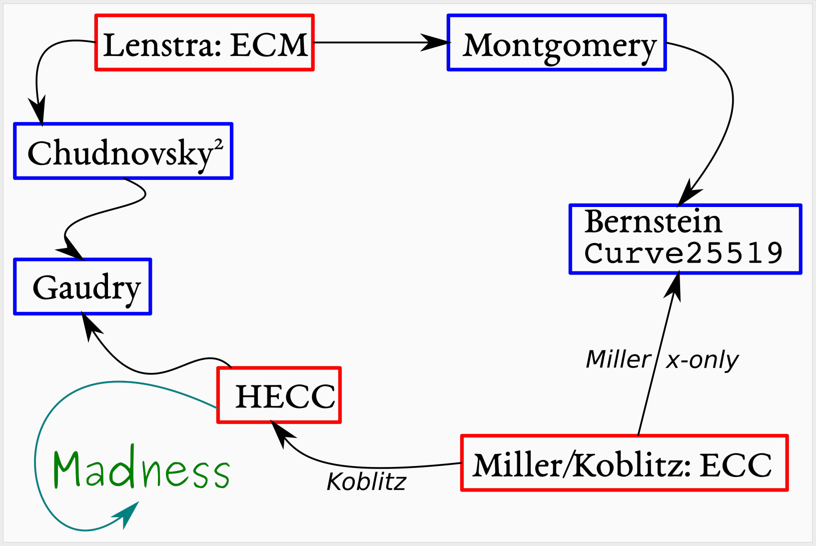
# Elliptic history: The missing link

In 2005, Pierrick Gaudry found out what would happen if you tried to do fast ECDH while reading the end of the Chudnovskys' paper instead of the end of Montgomery's paper.

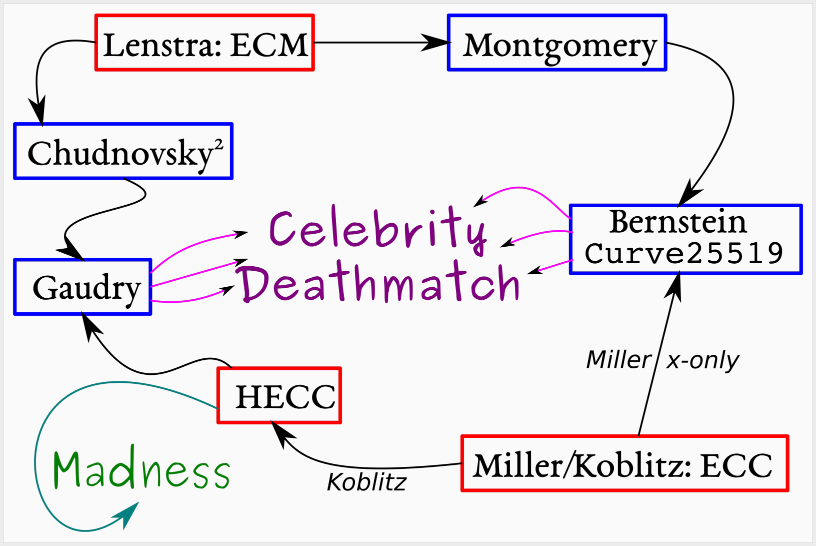
This made the missing link between the Chudnovskys' "abelian variety ECM" and genus-2 Diffie–Hellman.

*Later, Gaudry's student **Romain Cosset** tried using Kummer surfaces for ECM: fascinating, but not as useful as you'd hope.*

# Elliptic history: Return of the Chudnovskys



# Elliptic history: Things really get out of hand



# High-speed scalar multiplication

Bernstein–Chuengsatiansup–Lange–Schwabe (2014): high-speed Kummer implementations compete with elliptic scalar multiplication on both high- and low-spec platforms.

**Why** do Kummer surfaces beat elliptic $x$-only arithmetic when genus-2 Jacobians are *slower* than full elliptic curves?

*Part* of the answer is that we're using a **half-size finite field**.

But **symmetry** plays a huge role, too, by simplifying the polynomials that appear in the **pseudo-group operations**.

# Real-world efficiency improvements

For example, 256-bit-group scalar multiplication using $\mu$Kummer (Renes–Schwabe–S.–Batina) on 8-bit AVR ATmega:

| Diffie–Hellman | kCycles | Stack bytes |
|---|---|---|
| Curve25519 | 3590 | 548 |
| $\mu$Kummer | 2634 (73%) | 248 (45%) |

*On platforms with vector instructions we can do even better.*

# Signatures for microcontrollers

# Signatures for microcontrollers

Kummer surfaces are good for compact, fast **Diffie–Hellman**, but we also want **signatures**.

**Problem**: verifying signatures means checking equations like

$$R = [s]P + [e]Q$$

where $R$, $P$, $Q$ are in a group... and Kummer surfaces have no $+$.

*How can we expoit the speed of Kummer/Montgomery arithmetic for signatures?*

# Conventional approach: Don't do it.

**Don't do it.** Use Kummer/Montgomery for Diffie–Hellman, and a separate twisted Edwards curve for signatures.

**Example:** the NaCl library.

*Disadvantages:*

- slower arithmetic for signatures
- more stack space for Edwards coordinates
- two mathematical objects $\Longrightarrow$ bigger trusted code base
- two separate public key formats for DH and signatures

# Hybrid approach: Recovery

Use $\mathbb{P}^1$/Kummer for Diffie–Hellman. For signatures,

1. Start with group elements $P$ in the Jacobian;
2. Project $P$ to $\pm P$ on the Kummer, and compute $\pm[m]P$ there; the ladder actually computes $\pm[m]P$ **and** $\pm[m+1]P$, and the triple $(P, \pm[m]P, \pm[m+1]P)$ determines $[m]P$;
3. Point recovery formulæ give $[m]P$ back in the Jacobian, which has a full group law for signature verification.

**Pros:** Kummer speed for signatures.

**Cons:** bigger trusted code base (Kummer + Jacobian); mixed public key formats; recovery formulæ require a lot of stack space to compute.

# Putting the hybrid approach into practice

$\mu$Kummer (Renes–Schwabe–S.–Batina, CHES 2016):
An open crypto lib for 8- and 32-bit microcontrollers.
Efficient Diffie–Hellman *and* Schnorr signatures
using Kummer surfaces and genus-2 point recovery.

| | ATmega (8-bit) | | Cortex M0 (32-bit) | |
|---|---|---|---|---|
| | KCycles | Stack bytes | KCycles | Stack |
| DH | 9739 | 429 | 2644 | 584 |
| Keygen | 10206 | 812 | 2774 | 1056 |
| Sign | 10404 | 926 | 2865 | 1360 |
| Verify | 16241 | 992 | 4454 | 1432 |

**Pros:** faster and smaller than the elliptic SOA,
**Cons:** inconveniently large stack requirements.

# A new approach: qDSA

# Signature verification

All this (heavy) group stuff—twisted Edwards, Jacobians, point recovery—is only required because the signature verification

$$R = [s]P + [e]Q$$

requires a $+$, hence a group.

**Brutal solution**: instead, verify the slightly weaker relation

$$\pm R = \pm[s]P \pm [e]Q\,.$$

*Hamburg's elliptic Strobe library already (informally) does this!*

# quotient Digital Signature Algorithm

qDSA (Renes–S. 2017): a variant of EdDSA using *only* $\mathbb{P}^1$/Kummer arithmetic. *Cheap extension of Diffie–Hellman systems to provide signature schemes.*

Key pairs: Key pairs: $(\pm Q, x)$ such that $\pm Q = \pm[x]P$. Here $\pm Q$ is

- a Curve25519 key (elliptic version)
- a compressed Kummer point (genus-2 version)

Signatures: $(\pm R, s)$ with $\pm R \in \{\pm([s]P + [e]Q), \pm([s]P - [e]Q)\}$.

Pros: only fast Montgomery/Kummer arithmetic, unified public-key formats, and less stack space!

Cons: what cons?

# Checking $\pm R \in \{\pm([s]P \pm [e]Q)\}$

**To verify signatures**: classical theory of theta functions $\implies$ biquadratic polynomial equations in the coordinates of $\pm A$, $\pm B$ that are only satisified by the coordinates of $\pm(A \pm B)$.

**Elliptic version** on $\mathcal{E} : Y^2 Z = X(X^2 + cXZ + Z^2)$:

$$\pm R \in \left\{ \begin{array}{c} \pm(A+B)\,, \\ \pm(A-B) \end{array} \right\} \iff 2B_{XZ} \cdot X_R Z_R = B_{ZZ} \cdot X_R^2 + B_{XX} \cdot Z_R^2$$

where $B_{XX} = (X_A X_B - Z_A Z_B)^2$, $B_{ZZ} = (X_A Z_B - Z_A X_B)^2$,
and $B_{XZ} = (X_A X_B + Z_A Z_B)(X_A Z_B + Z_A X_B) + 2cX_A Z_A X_B Z_B$.

**Genus-2 version**: 10 biquadratic forms, 6 equations.

# Results

| | | ATmega (8-bit) | | Cortex M0 (32-bit) | |
|---|---|---|---|---|---|
| System | Function | Cycles | Stack | Cycles | Stack |
| Ed25519 | sign | 19048 | 1473 | — | — |
| | verify | 30777 | 1226 | — | — |
| Four$\mathbb{Q}$ | sign | 5175 | 1590 | — | — |
| | verify | 11468 | 5050 | — | — |
| qDSA-$\mathcal{E}$ | sign | 14070 | 412 | 3889 | 660 |
| | verify | 25375 | 644 | 6799 | 788 |
| $\mu$Kummer | sign | 10404 | 926 | 28635 | 1360 |
| | verify | 16240 | 992 | 4454 | 1432 |
| qDSA-$\mathcal{K}_\mathcal{C}$ | sign | 10477 | 417 | 2908 | 580 |
| | verify | 20423 | 609 | 5694 | 808 |

**Ed25519**: Nascimento et al (2015); **Four$\mathbb{Q}$**: Liu et al (2017);
**qDSA-$\mathcal{E}$**: qDSA/Curve25519; **qDSA-$\mathcal{K}_\mathcal{C}$**: qDSA/Gaudry–Schost Kummer