On being more than friendly

DiS lunch talk

Fabio Campos[1] Jorge Chavez-Saab[2] Jesús-Javier Chi-Domínguez[3]
Michael Meyer[4] Krijn Reijnders[6] Francisco Rodríguez-Henríquez[3]
Peter Schwabe[5] Thom Wiggers[6]

April, 2021

[1] University of Applied Sciences Wiesbaden, Germany
[2] Departamento de Computación, CINVESTAV-IPN, Mexico
[3] Cryptography Research Centre, TII, Abu Dhabi
[4] University of Regensburg, Germany
[5] Max Planck Institute for Security and Privacy, Bochum, Germany
[6] Radboud University Nijmegen, The Netherlands

Main ideas of the ongoing project

- performance evaluation of CSIDH[1,2] as Diffie-Hellman replacement
- post-quantum "CSIDH-OPTLS" vs post-quantum KEMTLS[3]
- optimized software for different variants of CSIDH
- high(er) security levels

---

[1] Commutative Supersingular Isogeny Diffie-Hellman
[2] https://eprint.iacr.org/2018/383.pdf
[3] https://eprint.iacr.org/2021/779

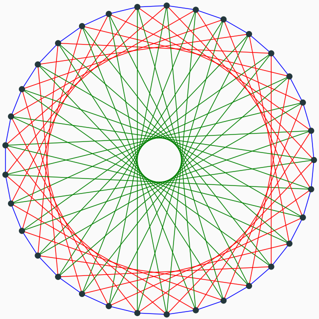
Outline

- CSIDH in a nutshell
- Modular multiplication
- Implementation on 64-bit Intel
- Current results

CSIDH in a nutshell

CSIDH (Commutative Supersingular Isogeny Diffie-Hellman)

- over $\mathbb{F}_p$ with $p = 4 \cdot \ell_1 \cdots \ell_n - 1$, s.t. $\ell_1, \ldots, \ell_n$ odd primes
- private key $= (e_1, \ldots, e_n)$, s.t. $|e_i| =$ number of isogenies of degree ℓ_i
- non-interactive key exchange $\rightsquigarrow$ drop-in replacement for Diffie-Hellman



supersingular curves over $\mathbb{F}_p$
$p = 659 = 4 \cdot 3 \cdot 5 \cdot 11 - 1$

CSIDH's Security

- **classical**: problem of finding a path $\simeq$ private key space
- **quantum**: relies on the size of prime p

Modular multiplication

Schoolbook and divide-and-conquer methods

Let $a = A_1 w + A_0$ and $b = B_1 w + B_0$

- **Schoolbook**: (4**M** + 1**A**)
  $a * b = A_1 B_1 w^2 + (A_0 B_1 + A_1 B_0) w + A_0 B_0$

- **Karatsuba**: (3**M** + 4**A**)
  $a * b = A_1 B_1 w^2 + ((A_0 + A_1)(B_0 + B_1) - A_1 B_1 - A_0 B_0) w + A_0 B_0$

- **Toom-Cook (*N*-way)**:
  splitting $A_n w^n + \cdots + A_1 w + A_0$ into N parts of n/N limbs

Montgomery reduction

Montgomery reduction

Algorithm 1: Montgomery reduction

Input : $p = (p_{n-1}, \ldots, p_1, p_0)_b$ w/ $\gcd(p, b) = 1$,
$r = b^n, p' = -p^{-1} \mod b$, and
$c = (c_{2n-1}, \ldots, c_1, c_0)_b < p * r$.

Output: $c * r^{-1} \mod p$.

```
A ← c, s.t. A = (a_{2n-1}, ..., a_1, a_0)
for i ∈ {0, ..., (n − 1)} do
    u_i ← a_i * p' mod b
    A ← A + u_i * p * b^i
A ← A / b^n
if A ≥ p then
    A ← A − p
return A
```

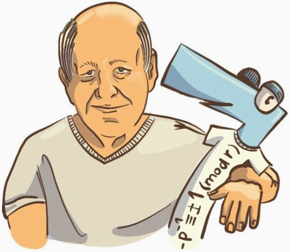
Montgomery friendly primes

Algorithm 2: Montgomery reduction

Input : $p = (p_{n-1}, \ldots, p_1, p_0)_b$ w/ $\gcd(p, b) = 1$,
and $p' = -p^{-1} = 1 \mod b, r = b^n$, and
$c = (c_{2n-1}, \ldots, c_1, c_0)_b < p * r$.

Output: $c * r^{-1} \mod p$.

$A \leftarrow c$, s.t. $A = (a_{2n-1}, \ldots, a_1, a_0)$
for $i \in \{0, \ldots, (n-1)\}$ **do**
    ~~$u_i \leftarrow a_i * p' \mod b$~~
    $A \leftarrow A + a_i * p * b^i$
$A \leftarrow A / b^n$
if $A \geq p$ **then**
    $A \leftarrow A - p$
return A



In Memoriam: Peter L. Montgomery (1947-2020)

Drawing by Lua Campos

More than friendly?

More than friendly?

Algorithm 3: Intermediate Montgomery reduction[a]

Input : e-bits long prime $p = 2^{e_2}\alpha - 1$ s.t. $e_2 \geq e/2$ and $0 \leq c < 2^e p$.

Output: $r_1 = c2^{-2e_2} \mod p$ and $0 \leq r_1 < p$.

$q_0 \leftarrow c \mod 2^{e_2}$

$r_0 \leftarrow (c - q_0)/2^{e_2} + q_0 * \alpha$

$q_1 \leftarrow r_0 \mod 2^{e_2}$

$r_1 \leftarrow (r0 - q_1)/2^{e_2} + q_1 * \alpha$

$r_1' \leftarrow r_1 - p + 2^e$

if $r_1' \geq 2^e$ **then**

  $r_1 \leftarrow r_1' \mod 2^e$

return r_1

[a] https://eprint.iacr.org/2020/665.pdf

In Memoriam: Peter L. Montgomery (1947-2020)

Implementation on 64-bit Intel

High level ideas

- dCSIDH and CTIDH sharing "constant-time" code
- optimized field arithmetic (MULX-Schoolbook, MULX-Karatsuba, GMP[4], AVX2)
- quantum security of CSIDH under debate[5]:

prime bits	key space	NIST level	Mont. reduction
p2048	2^{221}	1 (aggresive)	standard
p4096	2^{256}	1 (conservative)	standard
p5120	2^{234}	2 (aggresive)	intermediate
p6144	2^{256}	2 (conservative)	intermediate
p8192	2^{332}	3 (aggresive)	intermediate
p9216	2^{384}	3 (conservative)	intermediate

[4] https://gmplib.org/
[5] https://eprint.iacr.org/2020/1520

Low level ideas@AVX2

- Reduced-radix Representation (radix $= 26$)
- signed representation $\rightsquigarrow$ saving additions
- Karatsuba + schoolbook (operand scanning) at ≤ 12 limbs
- rollout of Karatsuba layers $\rightsquigarrow$ fewer multiplications
- interleaved multiplication

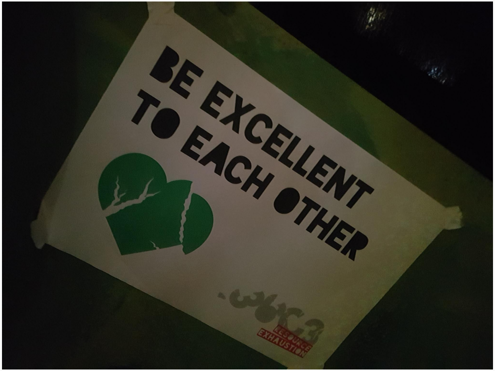
Current results

Current results and lessons learned

- AVX2 implementation is faster (not enough!)
- Intermediate reduction significantly faster
- CTIDH significantly faster than dCSIDH
- room for **many** wrong decisions
- More "theoretical" approach helpful?
  - operand vs product scanning?
  - number of Karatsuba layers?
  - interleaved vs non-interleaved multiplication?
  - which radix to choose?
  - signed vs unsigned?
  - ...
- @AVX2 carry handling ~~sucks~~ harder than expected

Thank you for your attention.

Let's be more than friendly.



picture @36th Chaos Communication Congress 2019