AprèsSQI: A Pretty Rad Extension to Signing in SQIsign

Maria Corte-Real Santos, **Jonathan Komada Eriksen**, Michael Meyer, Krijn Reijnders

Goals

- SQIsign:
  - Tiny signature + pk
  - Fast and easy verification
  - Slow signing
- Perfect for applications where signatures are verified many times.
- This work:
  - Accept that signing is going to be slow.
  - Make verification as fast as possible.

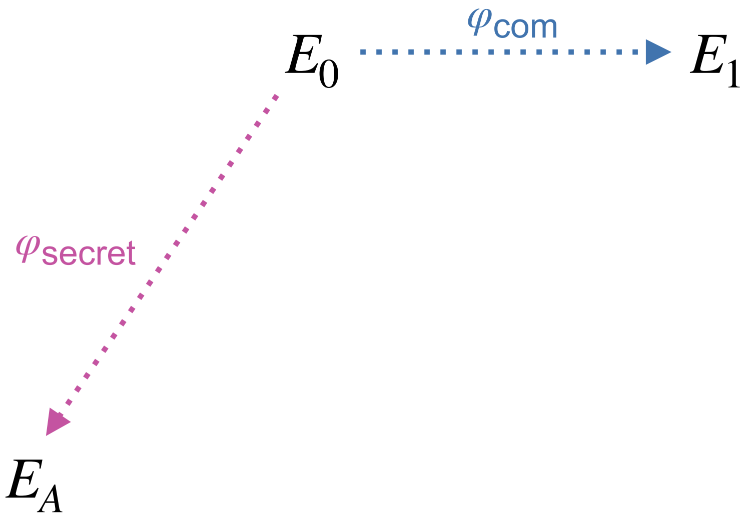
The Anatomy of a

SQISIGN SIGNATURE

SQIsign Signature - Key Generation

$$E_0 \xrightarrow{\varphi_{\text{secret}}} E_A$$

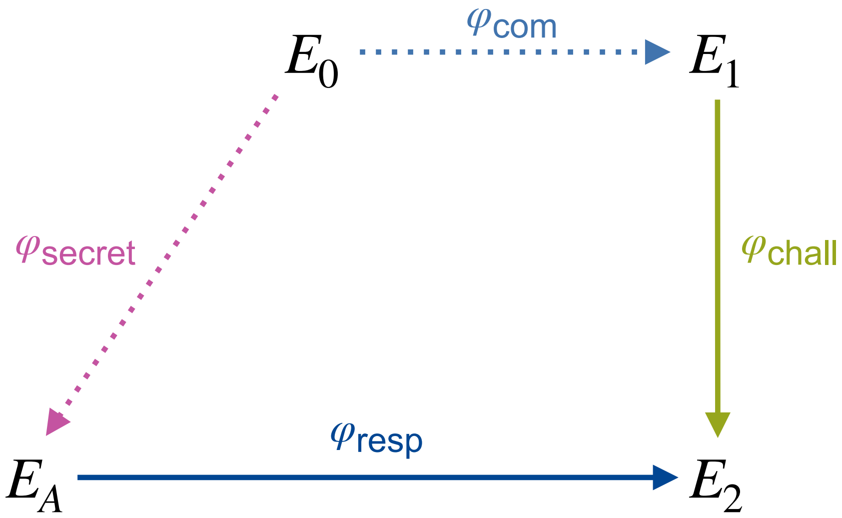
SQIsign Signature - Commitment



SQIsign Signature - Challenge

$$\begin{array}{ccc} E_0 & \xrightarrow{\varphi_{\text{com}}} & E_1 \\ \swarrow \varphi_{\text{secret}} & & \downarrow \varphi_{\text{chall}} \\ E_A & & E_2 \end{array}$$

SQIsign Signature - Response



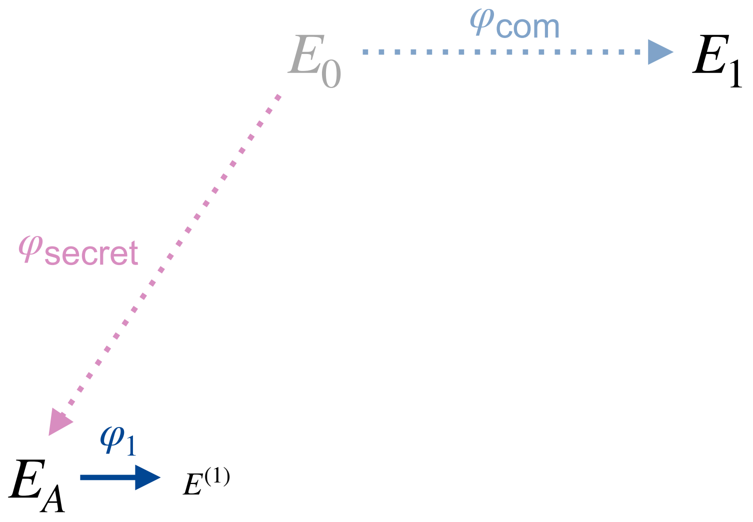
Uncompressed SQIsign Signature

- Verifying a signature on m:
  $Ver(m, \sigma, pk)$
  $\sigma = (\varphi_{\text{resp}}, E_1)$
- $\log(p) \approx 256$
- $2^{75} \mid\mid p + 1$
- $deg(\varphi_{\text{resp}}) \approx 2^{975}$
- $\varphi_{\text{resp}} : K_1, K_2, \ldots, K_{13}$

$$
\begin{tikzcd}
E_0 \arrow[r, dotted, "\varphi_{\text{com}}"] \arrow[d, dotted, "\varphi_{\text{secret}}"'] & E_1 \arrow[d, "\varphi_{\text{chall}}"] \\
E_A \arrow[r, "\varphi_{\text{resp}}"] & E_2
\end{tikzcd}
$$

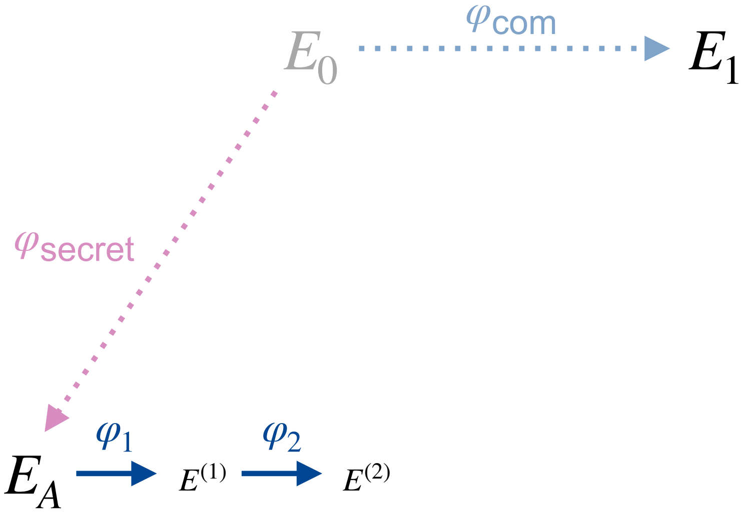
Uncompressed SQIsign Signature

- $\varphi_{\text{resp}} : K_1, K_2, \ldots, K_{13}$



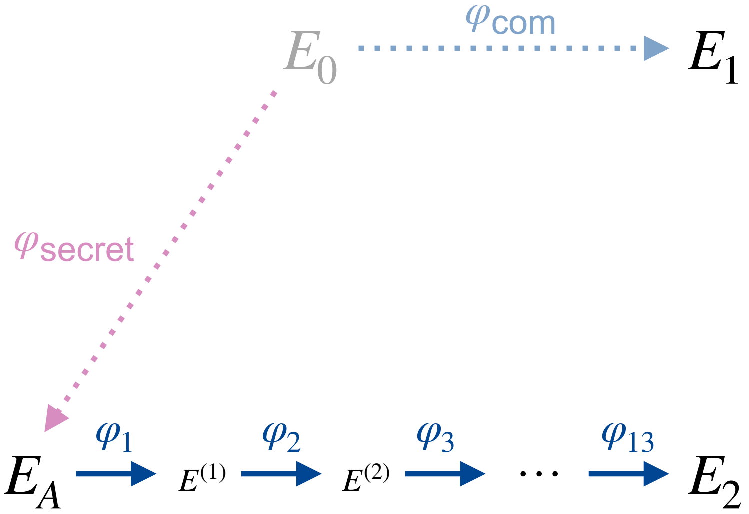
Uncompressed SQIsign Signature

- $\varphi_{\text{resp}} : K_1, K_2, \ldots, K_{13}$



Uncompressed SQIsign Signature

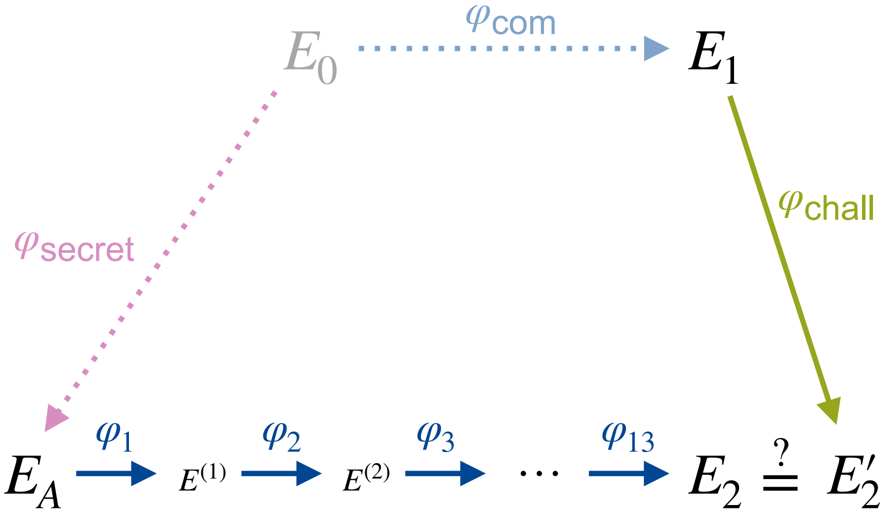
- $\varphi_{\text{resp}} : K_1, K_2, \ldots, K_{13}$



Uncompressed SQIsign Signature

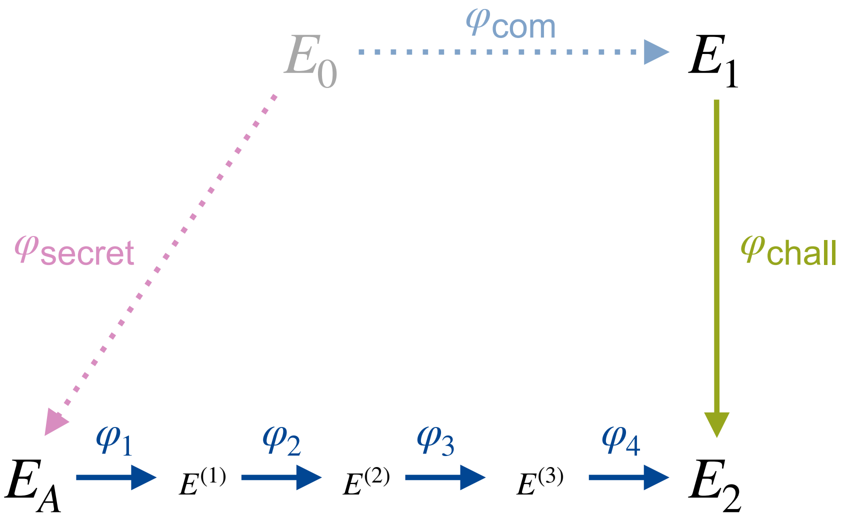
- $\varphi_{\text{resp}} : K_1, K_2, \ldots, K_{13}$
- $K_{\text{chall}} := H(E_1, m)$
- $\varphi_{\text{chall}} : E_1 \rightarrow E_2'$

$$\deg(\varphi_{chall}) = 2^{75}3^{36} > 2^{128}$$



Uncompressed SQIsign Signature

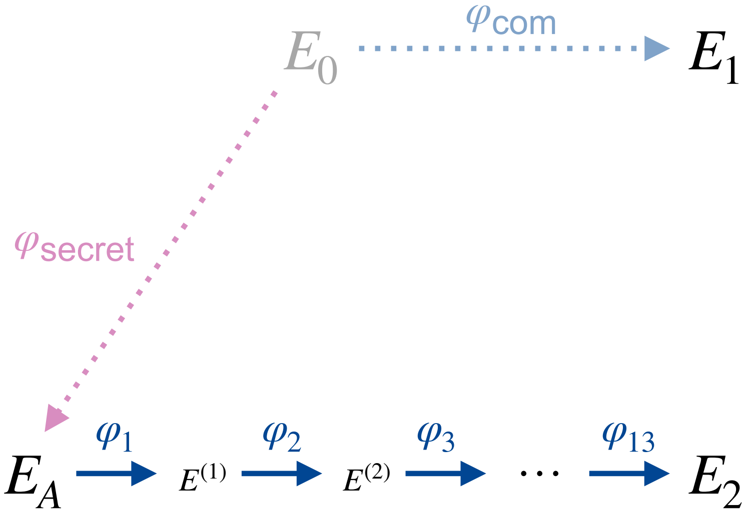
- $\varphi_{\text{resp}} : K_1, K_2, \ldots, K_{13}$
- $\lceil e/f \rceil = \lceil 975/75 \rceil = 13$ points in total
- Observation: Bigger f leads to smaller signatures.
- E.g. $f = 250$ gives 4 points.



Compressing the Response

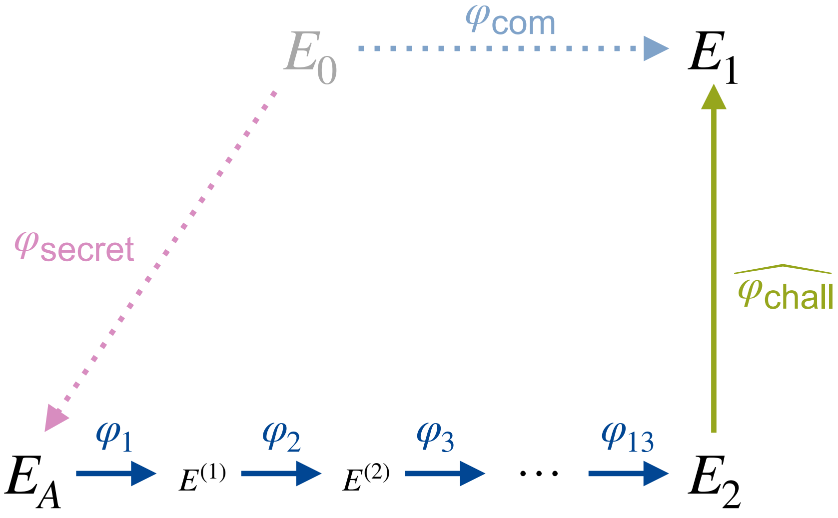
- $\varphi_{\text{resp}} : s_1, s_2, \ldots, s_{13}$
- $s_i \in \mathbf{Z}/2^f\mathbf{Z}$

- Each step:
  - Deterministically gen. $\langle P_i, Q_i \rangle = E^{(i-1)}[2^f]$
  - $K_i = P_i + [s_i]Q_i$
- Bigger $f \Rightarrow$ Faster



Compressing the Commitment

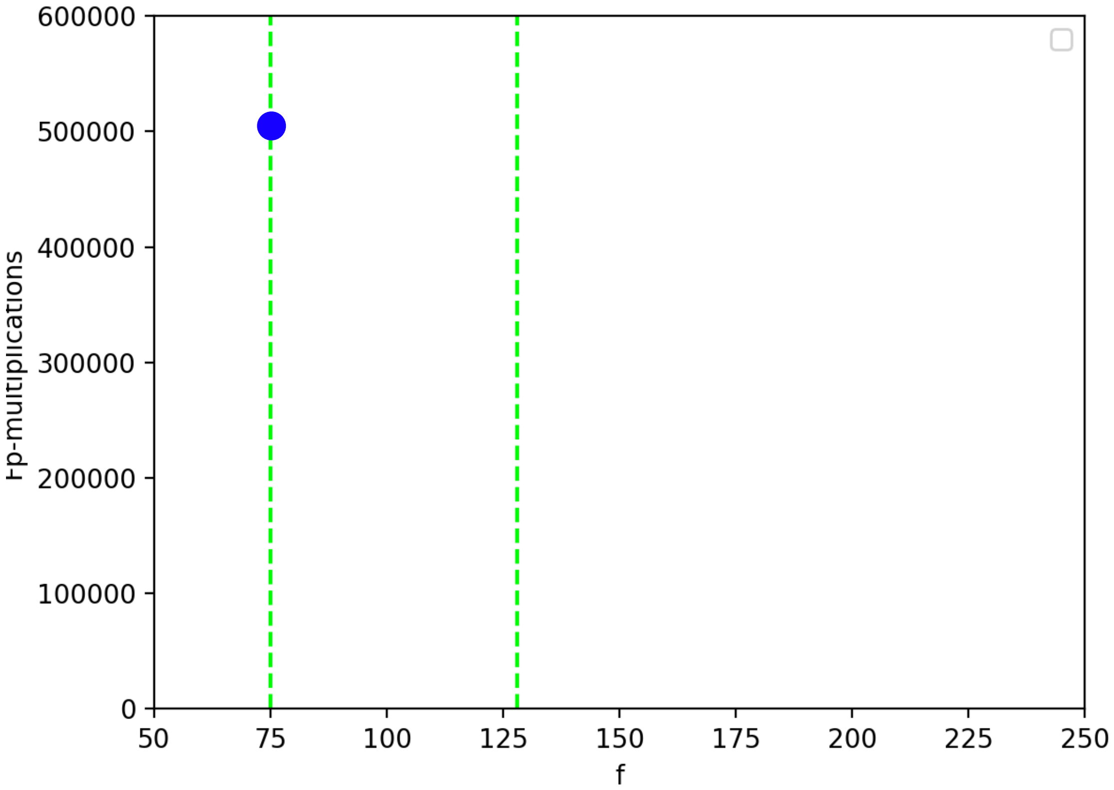
- Replace E_1 by (r, s_{chall})
- Deterministically gen. $\langle P, Q \rangle = E_2[D_{chall}]$
- $\widehat{\varphi_{chall}}$, generated by $P + [s_{chall}]Q$
- Verify: $\widehat{\varphi_{chall}}(Q) = [r]H(E_1, m)$



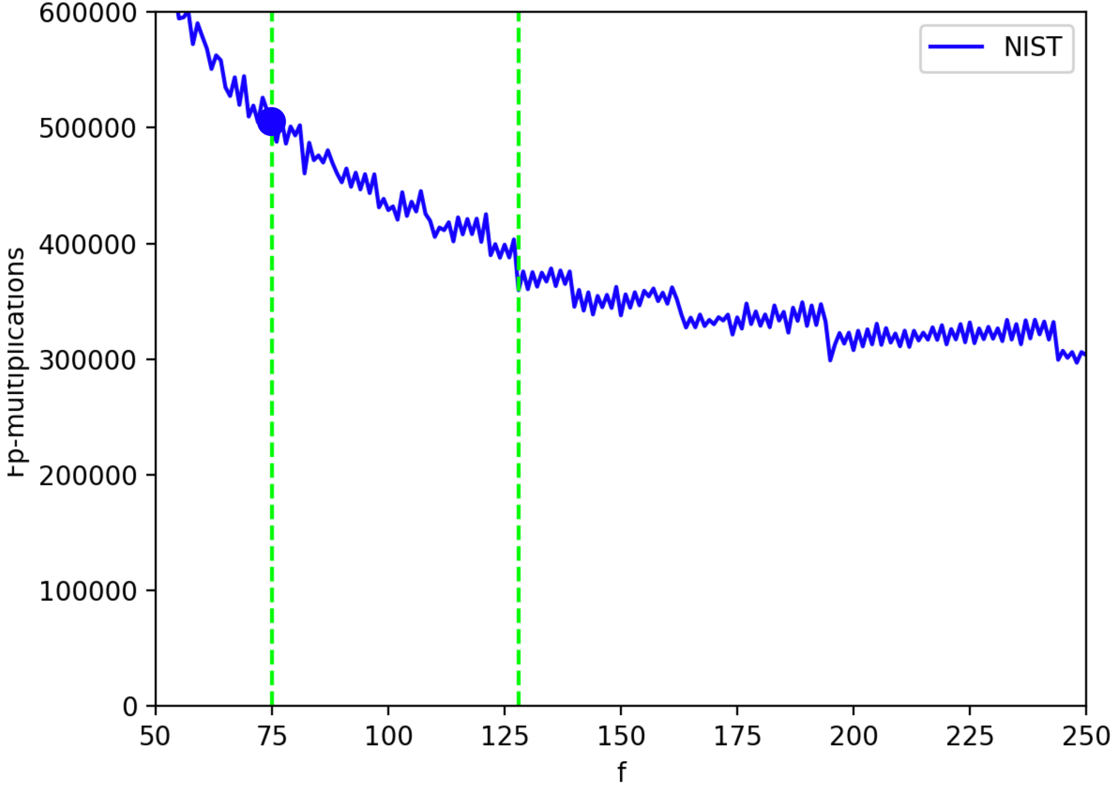
Resulting

BENCHMARKS

Effect of larger f



Effect of larger f



Other Optimisations

- Several low-level optimisations.
  - Faster basis generation.
  - Faster kernel point computation.
- Example: Given a curve

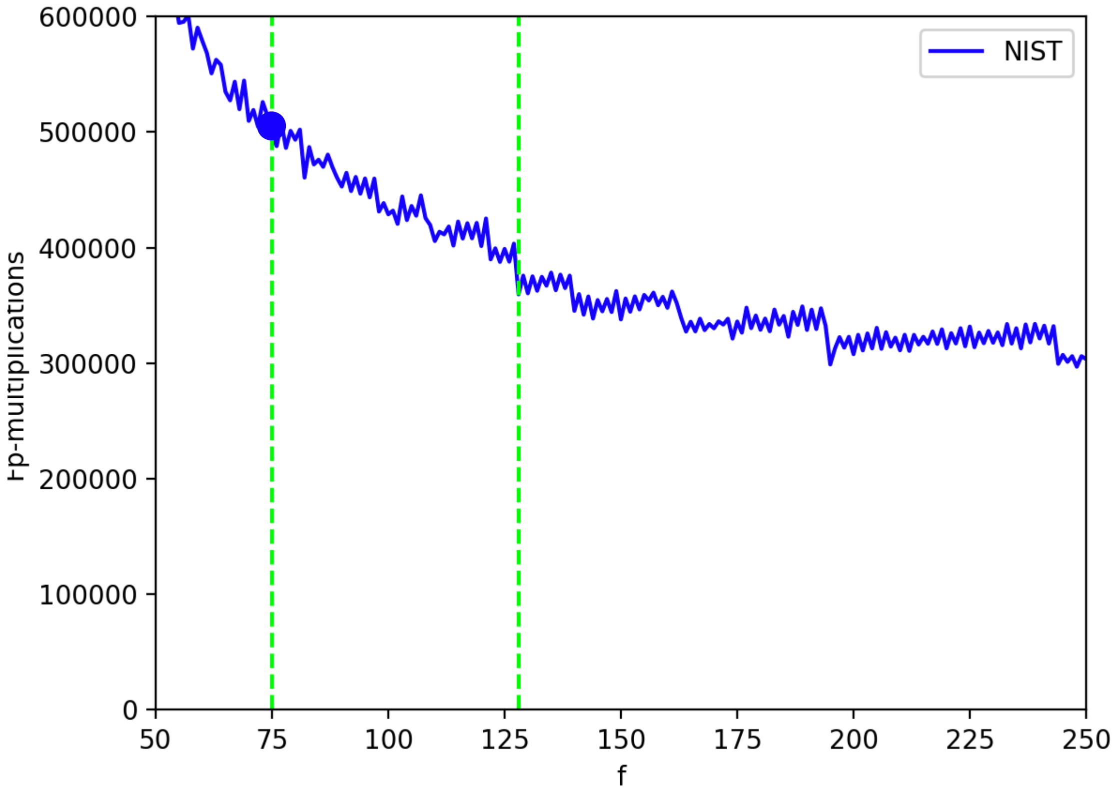
$$E : y^2 = (x - \lambda_1)(x - \lambda_2)(x - \lambda_3)$$

  - $Q \in [2]E$ iff. $x_Q - \lambda_i$ are square for all i.
  - $Q \in E \backslash [2]E$ "above" $(\lambda_i, 0) \in E[2]$ iff $x_Q - \lambda_i$ square

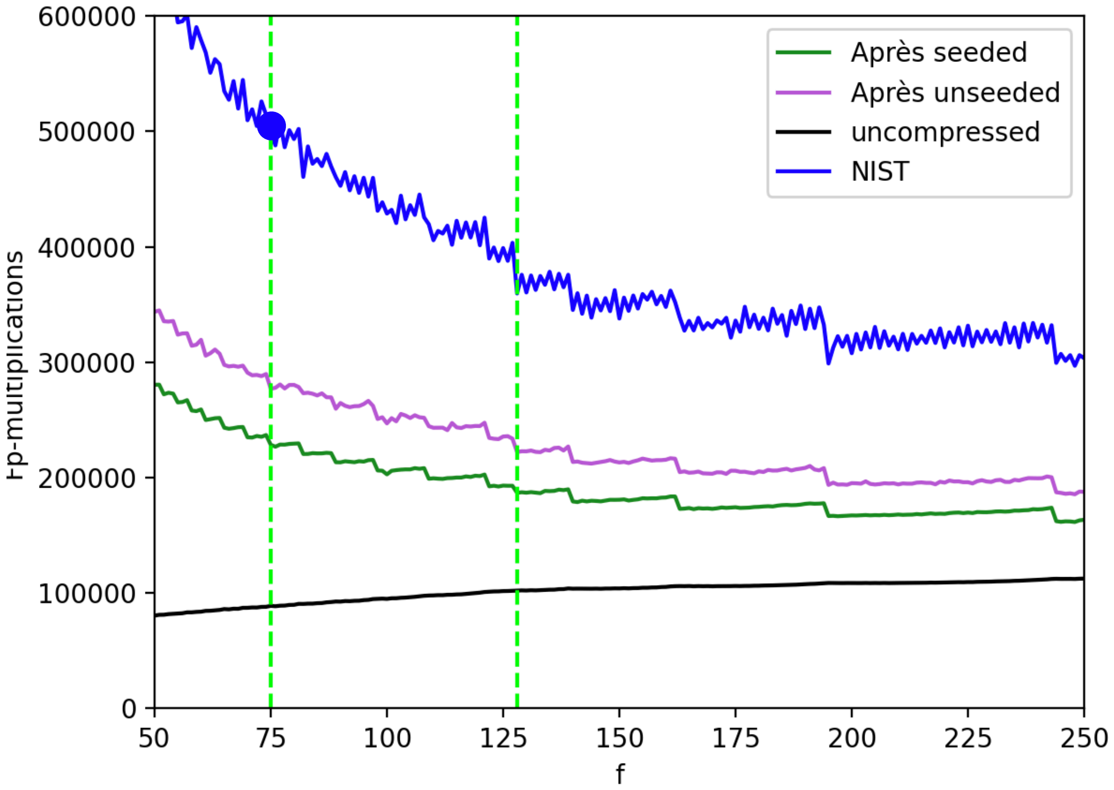
Size-speed trade-offs

- Re-introduce seeds.
  - Smaller and fewer seeds.
- Uncompressed signatures.
  - Compressed NIST-signatures: 177 B
  - Uncompressed NIST $(f = 75)$: 896 B
  - Uncompressed $f = 246$: 322 B

Effect of larger f

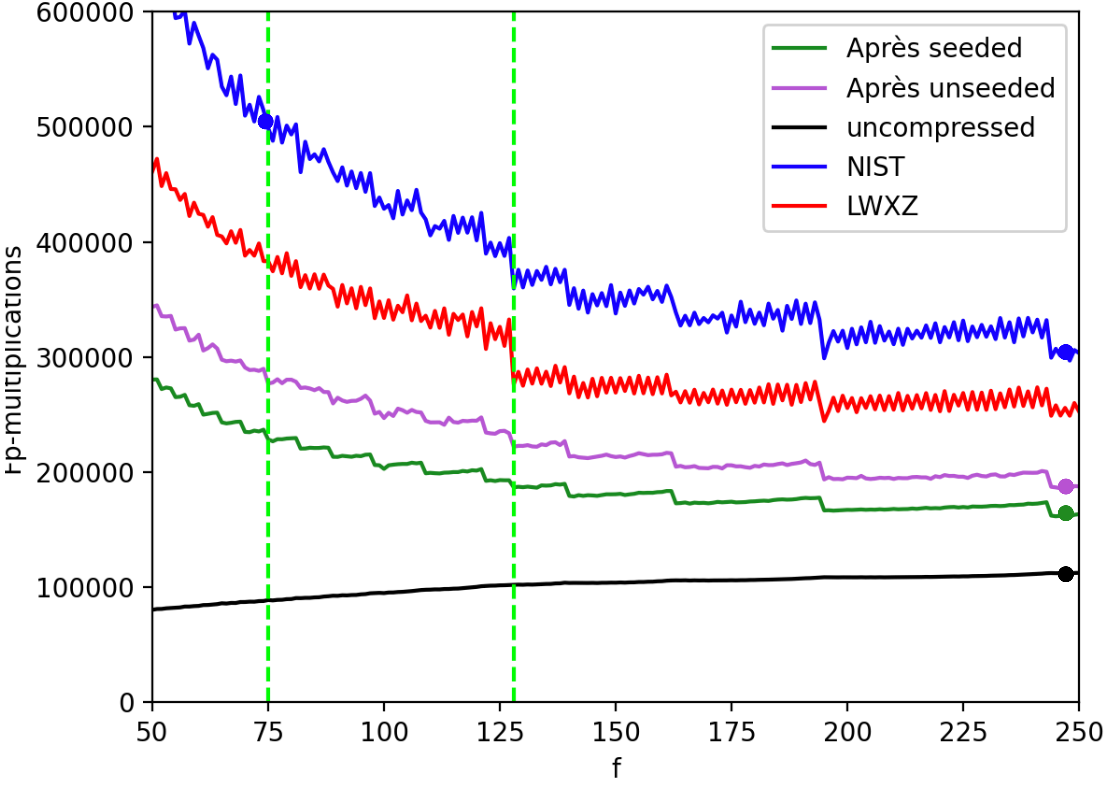


All Results



600000
500000
400000
300000
200000
100000
0
Fp-multiplications
50
75
100
125
150
175
200
225
250
f
Après seeded
Après unseeded
uncompressed
NIST

All Results



- Increasing f:
  - 1.68x faster
- Optimised:
  - 2.65x faster
- Seeded (+10 B)
  - 3.04x faster
- Uncomp. (2x B)
  - 4.40x faster

Obtaining larger f when

SIGNING WITH FIELD-EXTENSIONS

Current restrictions on f?

- SQIsign:
  - $2^f \mid p + 1$
  - $T \mid \cancel{(p^2 - 1)/2}$ odd, smooth and $T \approx p^{5/4}$

Changing the requirement on T

- Signing: Ideals of norm $T \Rightarrow$ isogenies of degree T.
    - Finding the kernel: Linear algebra + a few additions.
    - Computing isogeny from kernel points.
- SQIsign: All computations happen in $\mathbb{F}_{p^2}$
    - Hence $T \mid (p^2 - 1)/2$
- Allow kernel points to live in bigger extension-fields?
    - A few additions.
    - Computing rational isogeny from irrational generator.

Example prime

- 7-block verification

$$p_7 = 2^{145} \cdot 3^9 \cdot 59^3 \cdot 311^3 \cdot 317^3 \cdot 503^3 - 1.$$

T is 997-smooth

- 4-block verification

$$p_4 = 2^{242} \cdot 3 \cdot 67 - 1$$

T is 2293-smooth

$E(\mathbb{F}_{p^{2k}})$	Torsion group
$k=1$	$E[3^7], E[53^2], E[59^3], E[61], E[79], E[283], E[311^3]$ $E[317^3], E[349], E[503^2], E[859], E[997]$
$k=3$	$E[13], E[109], E[223], E[331]$
$k=4$	$E[17]$
$k=5$	$E[11], E[31], E[71], E[241], E[271]$
$k=6$	$E[157]$
$k=7$	$E[7^2], E[29], E[43], E[239]$
$k=8$	$E[113]$
$k=9$	$E[19^2]$
$k=10$	$E[5^4], E[41]$
$k=11$	$E[23], E[67]$
$k=12$	$E[193]$
$k=13$	$E[131]$
$k=15$	$E[181]$
$k=18$	$E[37], E[73]$
$k=23$	$E[47]$

Sage-Math Implementation

- Implementation available at github.com/TheSICQ/ApresSQI
- Builds on
  - NIST-documentation (thanks SQIsign-team!)
  - Learning to SQI (thanks Giacomo!)
  - Deuring for the People (thanks Lorenz, Jana and Mattia!)
- Proof-of-Concept SageMath implementation for comparison:

Table 1: Comparison between estimated cost of signing for three different primes.

p	largest $\ell \mid T$	largest $\mathbb{F}_{p^{2k}}$	$\textsc{SigningCost}_p(T)$	Adj. Cost	Timing
p_{1973}	1973	$k = 1$	8371.7	1956.5	15m, 53s
p_7	997	$k = 23$	4137.9	-	10m, 06s
p_4	2293	$k = 53$	9632.7	-	16m, 13s

- Optimised AprèsSQI signing competitive with current SQIsign signing?

New Prime Search Techniques?

- Change in requirement
  - $T \mid (p^2 - 1)/2$ ✗
  - $T \mid N$, where

$$N = \prod_{n=1}^{k} \Phi_n(p^2)/2 \quad ✓$$

and Φ_n denotes the n-th cyclotomic polynomial.

- N grows quickly with k, intuitively, a lot easier
  - How to best exploit this?


THANK YOU!