A Comparison of Open-Source Homomorphic Libraries With Multi-Precision Plaintext Moduli

Carlos Aguilar-Melchor Marc-Olivier Killijian
Cédric Lefebvre Tancrède Lepoint Thomas Ricosset

Université de Toulouse, INP-ENSEEIHT, CNRS, IRIT

Thales Communications & Security

WHEAT 2016

CYCLOTOMIC POLYNOMIAL RINGS

m-th cyclotomic polynomial:

$$\Phi_m(X) := \prod_{k \in \mathbb{Z}_m^*} \left(x - e^{2\pi i \frac{k}{m}} \right)$$

$$n := \deg(\Phi_m(X)) = \phi(m)$$

m-th cyclotomic polynomial ring of integers modulo q:

$$\mathcal{R}_q := \mathbb{Z}_q[X]/\Phi_m(X)$$

HELIB VARIANT OF BGV [HALEVISHOUP12,BGV12]

Key Generation

$$\mathbf{s} \xleftarrow{\chi_s} \mathcal{R}_{q_0}$$
$$\mathbf{a} \xleftarrow{\$} \mathcal{R}_{q_0}$$
$$\mathbf{e} \xleftarrow{\chi_e} \mathcal{R}_{q_0}$$

$$\mathbf{b} = -\mathbf{a} \cdot \mathbf{s} + p\mathbf{e}$$

$$\vec{s} = \begin{pmatrix} \mathbf{1} \\ \mathbf{s} \end{pmatrix}$$
$$\vec{k} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix}$$
$$\vec{r} \approx \mathsf{Enc}\left(\mathbf{s}^2\right)$$

Encryption

$$\boldsymbol{\mu} \in \mathcal{R}_p$$
$$\vec{k} \in \mathcal{R}_{q_0}^2$$
$$\mathbf{e_0}, \mathbf{e_1} \xleftarrow{\chi_e} \mathcal{R}_{q_0}$$

$$\mathbf{c_0} = \mathbf{b} + p\mathbf{e_0} + [q_0\boldsymbol{\mu}]_p$$
$$\mathbf{c_1} = \mathbf{a} + p\mathbf{e_1}$$

$$\vec{c} = \begin{pmatrix} \mathbf{c_0} \\ \mathbf{c_1} \end{pmatrix}$$

Decryption

$$\vec{c} \in \mathcal{R}_{q_i}^2$$
$$\vec{s} \in \mathcal{R}_{q_i}^2$$

$$\boldsymbol{d} = [\langle \vec{c}, \vec{s} \rangle]_{q_i}$$
$$= \mathbf{c_0} + \mathbf{c_1} \cdot \mathbf{s}$$

$$\boldsymbol{\mu} = \left[q_i^{-1}\mathbf{d}\right]_p$$

BGV: HOMOMORPHIC OPERATIONS

Addition

$$\vec{c_0} \in \mathcal{R}^2_{q_i} \qquad \vec{c_1} \in \mathcal{R}^2_{q_i} \qquad \vec{c} = [\vec{c_0} + \vec{c_1}]_{q_i} \qquad \vec{c} \in \mathcal{R}^2_{q_i}$$

Multiplication

$$\vec{c_0} = \begin{pmatrix} \mathbf{c_{00}} \\ \mathbf{c_{01}} \end{pmatrix} \qquad \vec{c_1} = \begin{pmatrix} \mathbf{c_{10}} \\ \mathbf{c_{11}} \end{pmatrix}$$

$$\mathbf{c_0} = \mathbf{c_{00}} \cdot \mathbf{c_{10}}$$
$$\mathbf{c_1} = \mathbf{c_{00}} \cdot \mathbf{c_{11}} + \mathbf{c_{01}} \cdot \mathbf{c_{10}}$$
$$\mathbf{c_2} = \mathbf{c_{01}} \cdot \mathbf{c_{11}}$$

$$\vec{c} = \begin{pmatrix} [q_i^{-1}]_p \mathbf{c_0} \\ [q_i^{-1}]_p \mathbf{c_1} \\ [q_i^{-1}]_p \mathbf{c_2} \end{pmatrix}$$

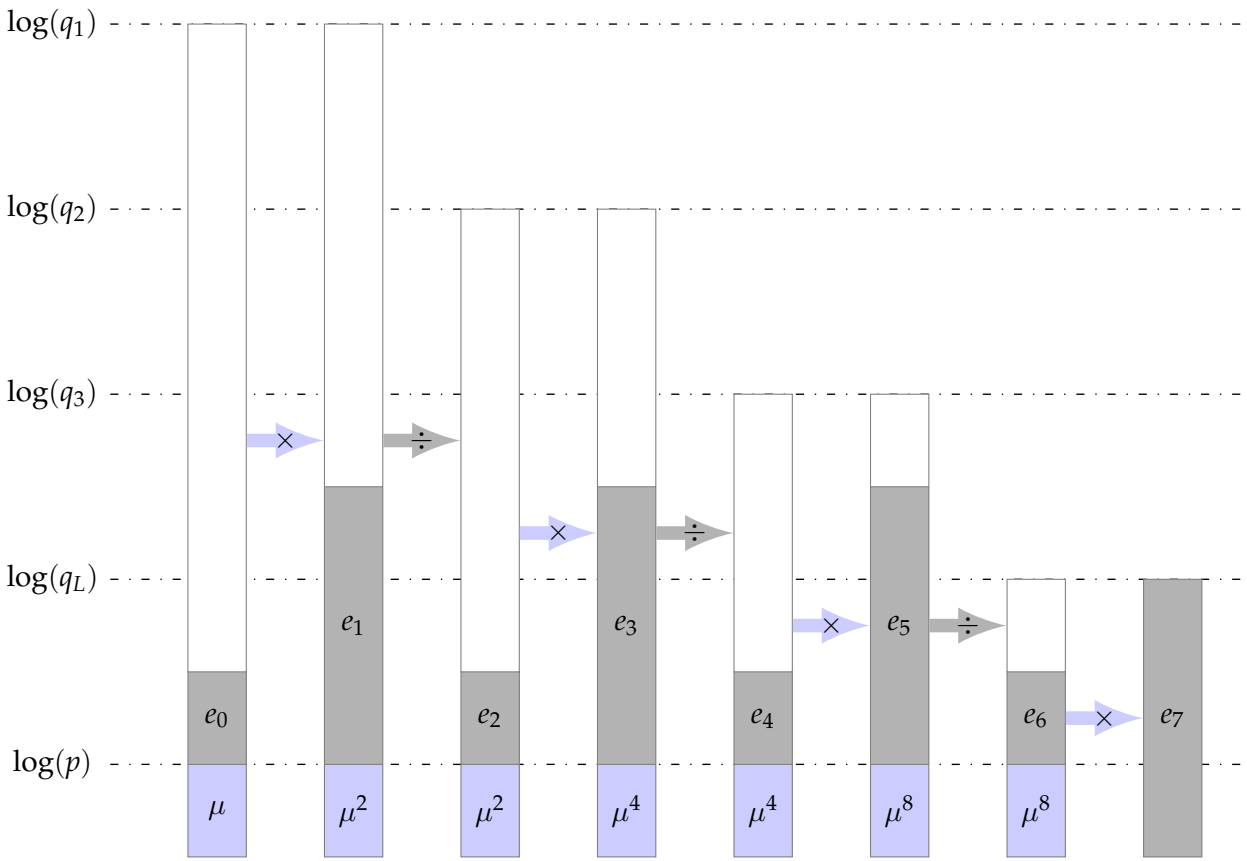
Relinearisation

$$\vec{c} = \begin{pmatrix} \mathbf{c_0} \\ \mathbf{c_1} \\ \mathbf{c_2} \end{pmatrix}$$

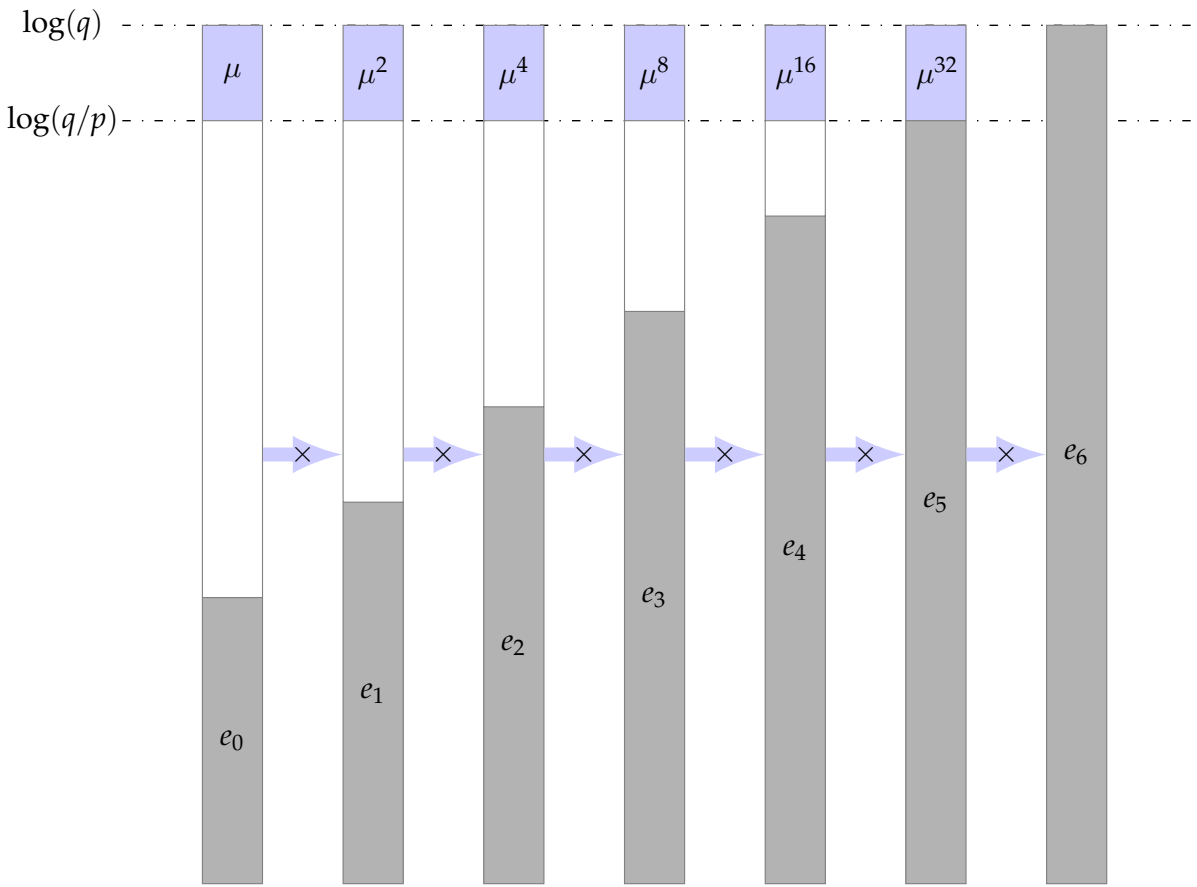
$$\tilde{\mathbf{c}}_\mathbf{i} = \sum_j \mathsf{Dcp}_j(\mathbf{c_2}) \cdot \mathsf{Pow}_j(\mathbf{r_i})$$
$$\mathbf{c'_0} = \mathbf{c_0} + \tilde{\mathbf{c}}_\mathbf{0}$$
$$\mathbf{c'_1} = \mathbf{c_1} + \tilde{\mathbf{c}}_\mathbf{1}$$

$$\vec{c'} = \begin{pmatrix} \mathbf{c'_0} \\ \mathbf{c'_1} \end{pmatrix}$$

BGV: MODULUS SWITCHING



SCALE INVARIANCE [BRAKERSKI12]



FV [FANVERCAUTEREN12]

Key Generation

$$\mathbf{a} \xleftarrow{\$} \mathcal{R}_{q_0}$$
$$\mathbf{e}, \mathbf{s} \xleftarrow{\chi_e} \mathcal{R}_{q_0}$$

$$\mathbf{b} = -\mathbf{a} \cdot \mathbf{s} + \mathbf{e}$$

$$\vec{s} = \begin{pmatrix} \mathbf{1} \\ \mathbf{s} \end{pmatrix}$$
$$\vec{k} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix}$$
$$\vec{r} \approx \mathsf{Enc}\left(\mathbf{s}^2\right)$$

Encryption

$$\boldsymbol{\mu} \in \mathcal{R}_p$$
$$\vec{k} \in \mathcal{R}_{q_0}^2$$
$$\mathbf{u}, \mathbf{e_1}, \mathbf{e_2} \xleftarrow{\chi_e} \mathcal{R}_{q_0}$$

$$\mathbf{c_0} = \mathbf{b} \cdot \mathbf{u} + \mathbf{e_1} + \left\lfloor \frac{q}{p} \right\rfloor \boldsymbol{\mu}$$
$$\mathbf{c_1} = \mathbf{a} \cdot \mathbf{u} + \mathbf{e_2}$$

$$\vec{c} = \begin{pmatrix} \mathbf{c_0} \\ \mathbf{c_1} \end{pmatrix}$$

Decryption

$$\vec{c} \in \mathcal{R}_{q_i}^2$$
$$\vec{s} \in \mathcal{R}_{q_i}^2$$

$$\mathbf{d} = \left[\langle \vec{c}, \vec{s} \rangle\right]_{q_i}$$
$$= \mathbf{c_1} + \mathbf{c_2} \cdot \mathbf{s}$$

$$\boldsymbol{\mu} = \left[\left\lfloor \frac{p\mathbf{d}}{q} \right\rceil \right]_p$$

FV: HOMOMORPHIC OPERATIONS

Addition

$$\vec{c_0} \in \mathcal{R}_{q_i}^2 \qquad \vec{c_1} \in \mathcal{R}_{q_i}^2 \qquad \vec{c} = [\vec{c_0} + \vec{c_1}]_{q_i} \qquad \vec{c} \in \mathcal{R}_{q_i}^2$$

Multiplication

$$\vec{c_0} = \begin{pmatrix} \mathbf{c_{00}} \\ \mathbf{c_{01}} \end{pmatrix} \qquad \vec{c_1} = \begin{pmatrix} \mathbf{c_{10}} \\ \mathbf{c_{11}} \end{pmatrix}$$

$$\mathbf{c_0} = \mathbf{c_{00}} \cdot \mathbf{c_{10}}$$
$$\mathbf{c_1} = \mathbf{c_{00}} \cdot \mathbf{c_{11}} + \mathbf{c_{01}} \cdot \mathbf{c_{10}}$$
$$\mathbf{c_2} = \mathbf{c_{01}} \cdot \mathbf{c_{11}}$$

$$\vec{c} = \begin{pmatrix} [\lfloor p\mathbf{c_0}/q \rceil]_q \\ [\lfloor p\mathbf{c_1}/q \rceil]_q \\ [\lfloor p\mathbf{c_2}/q \rceil]_q \end{pmatrix}$$

Relinearisation

$$\vec{c} = \begin{pmatrix} \mathbf{c_0} \\ \mathbf{c_1} \\ \mathbf{c_2} \end{pmatrix}$$

$$\tilde{\mathbf{c}}_\mathbf{i} = \sum_j \mathsf{Dcp}_j(\mathbf{c_2}) \cdot \mathsf{Pow}_j(\mathbf{r_i})$$
$$\mathbf{c'_0} = \mathbf{c_0} + \tilde{\mathbf{c}}_\mathbf{0}$$
$$\mathbf{c'_1} = \mathbf{c_1} + \tilde{\mathbf{c}}_\mathbf{1}$$

$$\vec{c'} = \begin{pmatrix} \mathbf{c'_0} \\ \mathbf{c'_1} \end{pmatrix}$$

DOUBLE-CRT REPRESENTATION

For $\mathbf{a} \in \mathcal{R}_q$

Chinese Remainder Theorem (or Residue Number System):

$$\mathsf{CRT}(\mathbf{a}) := (\mathbf{a} \bmod p_i)_{1 \le i \le k}$$
$$\in \mathcal{R}_{p_1} \times \mathcal{R}_{p_2} \times \cdots \times \mathcal{R}_{p_k} \cong \mathcal{R}_q$$

Number Theoretic Transform:

$$\mathsf{NTT}(\mathbf{a}) := \left(\mathbf{a}(\zeta^j)\right)_{j \in \mathbb{Z}_m^*}$$
$$\in \mathbb{Z}_q^n \cong \mathcal{R}_q$$

Double-CRT representation:

$$\mathsf{doubleCRT}(\mathbf{a}) := \left(\mathbf{a}(\zeta_i^j) \bmod p_i\right)_{\substack{1 \le i \le k \\ j \in \mathbb{Z}_m^*}}$$
$$\in \mathbb{Z}_{p_1}^n \times \mathbb{Z}_{p_2}^n \times \cdots \times \mathbb{Z}_{p_k}^n \cong \mathcal{R}_q$$

HELIB

Uses NTL [Shoup10] and generic Bluestein's NTT

- optimum parameters can be selected
- slower than specific algorithms

Implements BGV with many optimizations

- full CRT homomorphic operations [GHS12] (only NTTs)
- begin with large q_0 and modulus switch to limit the noise

FV-NFLLIB

Uses NFLlib [ABGGKL16] with optimized NTT

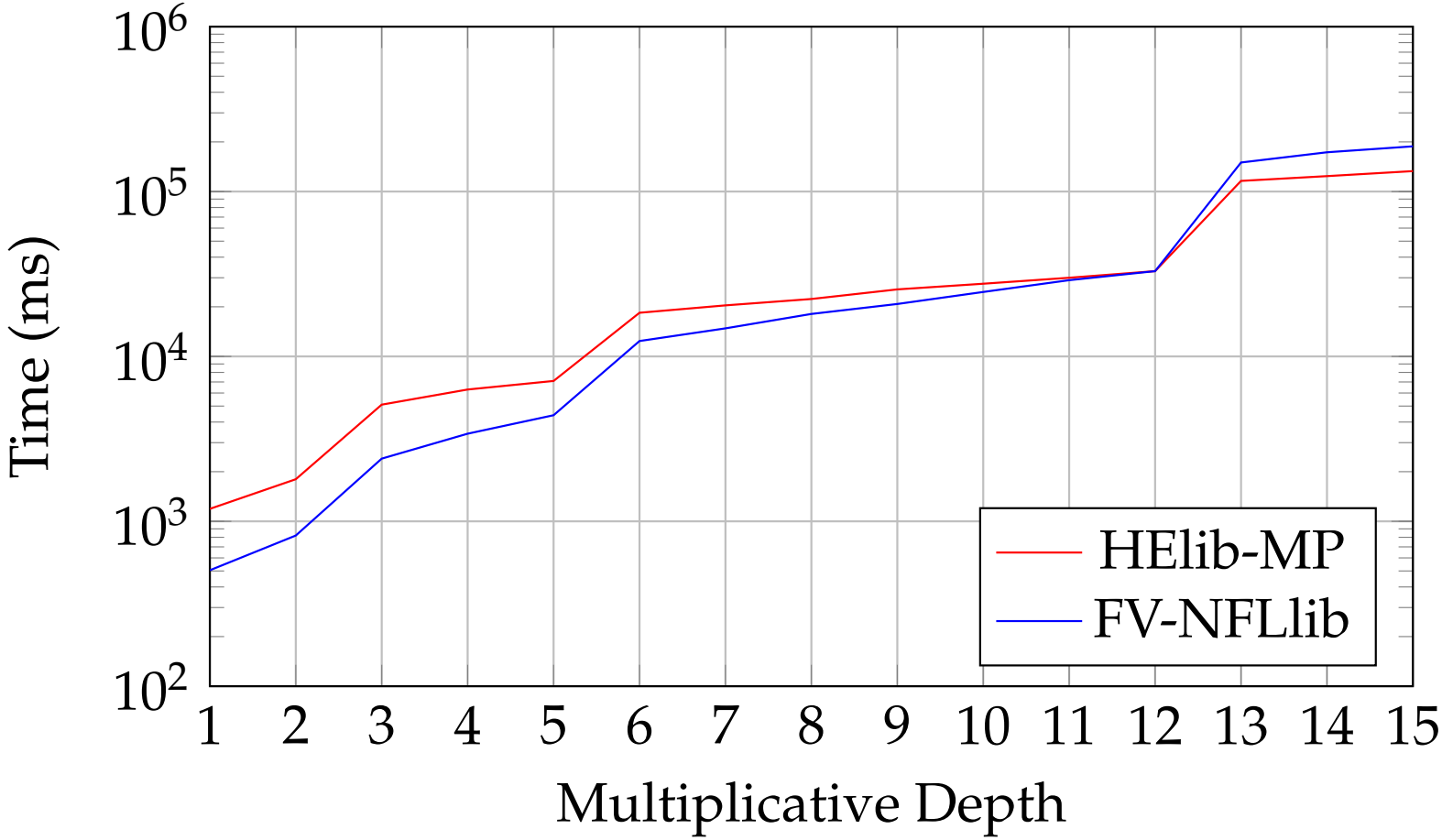
- faster than generic algorithms
- m must be a power of two

Implements FV with minimal optimization

- fixed q lower than for BGV
- CRT conversions needed for multiplication: $O(n \log(q)^2)$

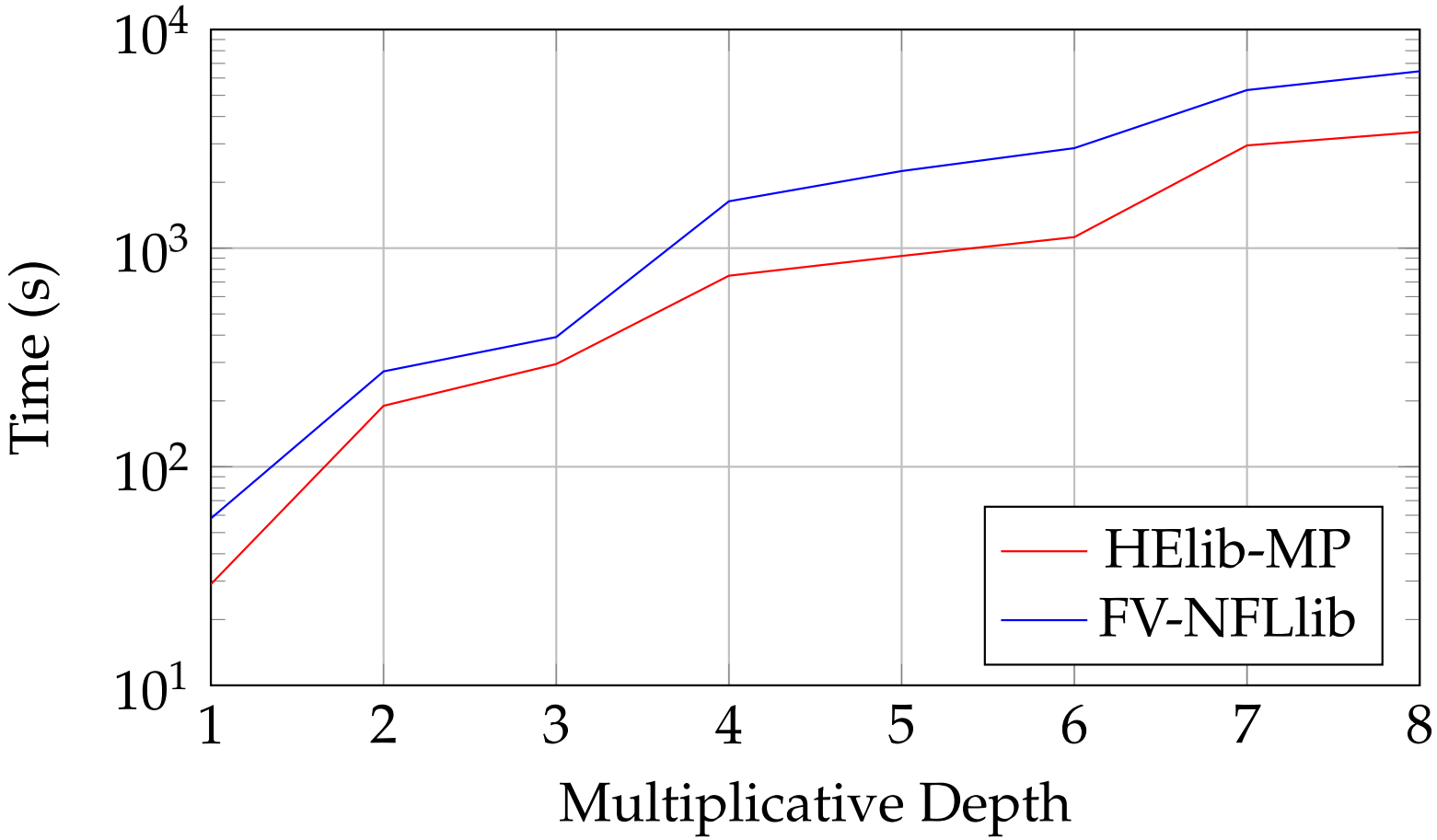
256-BIT PLAINTEXT

Average time for one multiplication ($\lambda > 80$ [APS15]) ran on one core of an Intel® Xeon™ E5-2695 v3 @ 2.30GHz



2048-BIT PLAINTEXT

Average time for one multiplication ($\lambda > 80$ [APS15]) ran on one core of an Intel® Xeon™ E5-2695 v3 @ 2.30GHz



HOMOMORPHIC MODULAR EXPONENTIATION

Compute $c^d \mod N$ with secret exponent $d = (d_0, \ldots, d_{\ell-1})_{2^\omega}$:

1. compute in clear all the possible sub-exponentiations:
$$\left(c_{i,j} = c^{i \cdot 2^{\omega \cdot j}} \mod N\right)_{\substack{0 \le i \le 2^\omega - 1 \\ 0 \le j \le \ell - 1}}$$
2. get a tuple of PIR queries $(\Omega_0, \ldots, \Omega_{\ell-1})$:
$$\Omega_j = \left(\mathsf{Enc}(\delta_{i,d_j})\right)_{0 \le i \le 2^\omega - 1}$$
3. homomorphically compute each window:
$$\mathsf{Enc}(c^{d_j \cdot 2^{\omega \cdot j}}) = \sum_{i=0}^{2^\omega - 1} c_{i,j} \cdot \mathsf{Enc}(\delta_{i,d_j})$$
4. homomorphically compute the exponentiation:
$$\mathsf{Enc}(c^d) = \prod_{j=0}^{\ell-1} \mathsf{Enc}(c^{d_j \cdot 2^{\omega \cdot j}})$$

BENCHMARK: RSA-2048 WITH HELIB-MP ($\lambda > 128$)

Window Size	Mult. Depth	Size of q_0	n	# Mult.	Time per Exp.
8	1	8.5 kb	3×10^5	128	157ms
8	2	13 kb	4×10^5	192	237ms
8	3	17 kb	5×10^5	224	377ms
8	4	21 kb	6×10^5	240	555ms
8	5	26 kb	7×10^5	248	687ms
8	6	30 kb	8×10^5	252	904ms
8	7	34 kb	9×10^5	254	1.1s
8	8	38 kb	1×10^6	255	1.8s
16	1	8.5 kb	3×10^5	64	78ms
16	2	13 kb	4×10^5	96	122ms
16	3	17 kb	5×10^5	112	187ms
16	4	21 kb	6×10^5	120	270ms
16	5	26 kb	7×10^5	124	347ms
16	6	30 kb	8×10^5	126	546ms
16	7	34 kb	9×10^5	127	550ms

ran on one core of an Intel® Core™ i5-4210H @ 2.90GHz

BENCHMARK: RSA-2048 WITH HELIB-MP ($\lambda > 128$)

Window Size	Mult. Depth	Size of q_0	n	# Mult.	Time per Exp.	Time
8	1	8.5 kb	3×10^5	128	157ms	13h
8	2	13 kb	4×10^5	192	237ms	26h
8	3	17 kb	5×10^5	224	377ms	52h
8	4	21 kb	6×10^5	240	555ms	93h
8	5	26 kb	7×10^5	248	687ms	134h
8	6	30 kb	8×10^5	252	904ms	201h
8	7	34 kb	9×10^5	254	1.1s	276h
8	8	38 kb	1×10^6	255	1.8s	383h
16	1	8.5 kb	3×10^5	64	78ms	6.5h
16	2	13 kb	4×10^5	96	122ms	13h
16	3	17 kb	5×10^5	112	187ms	26h
16	4	21 kb	6×10^5	120	270ms	45h
16	5	26 kb	7×10^5	124	347ms	67h
16	6	30 kb	8×10^5	126	546ms	121h
16	7	34 kb	9×10^5	127	550ms	137h

ran on one core of an Intel® Core™ i5-4210H @ 2.90GHz

ECC-ELGAMAL-P256 WITH HELIB-MP ($\lambda > 128$)

- Complete addition using [RCB15, Algorithm 4]:
  3 coordinates, mult. depth 2 and 12 multiplications

Window Size	Mult. Depth	Size of q_0	n	# Elliptic Curve Add.	Time per EC Mul.	Time
8	10	7.5 kb	2.2×10^5	31	242ms	14.5h
16	8	6 kb	1.8×10^5	15	96ms	5h

ran on one core of an Intel® Core™ i5-4210H @ 2.90GHz


Thank you!

Binary Plaintext