清華大學
Tsinghua University

Accelerating BGV Bootstrapping for Large p Using Null Polynomials over $\mathbb{Z}_{p^e}$

Shihe Ma, Tairong Huang, Anyu Wang, Xiaoyun Wang

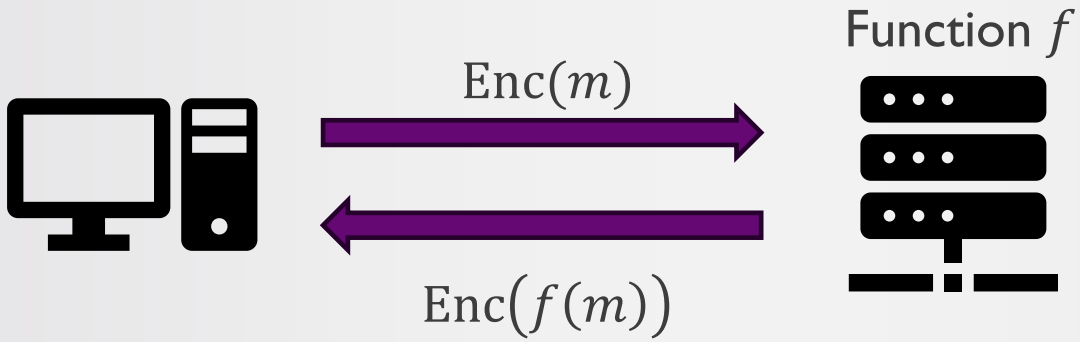
Tsinghua University

Overview

- Background on FHE and BGV
- Existing BGV bootstrapping: overview and limitations
- Our method and contribution
  - Method 1: Local null polynomials
  - Method 2: Global null polynomials
  - Experimental results
- Conclusion

Concept of FHE

- Fully Homomorphic Encryption(FHE) allows anyone to compute on encrypted data without access to the decryption key.
- The concept was first proposed by Rivest, Adleman, and Dertouzos in 1978
- The first FHE scheme was designed by Gentry in 2009



The BGV Scheme

- RLWE-based FHE scheme designed by Brakerski, Gentry, and Vaikuntanathan in 2011
- $\mathcal{R} = \mathbb{Z}[X]/\Phi_M(X), \mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for some q
- Plaintext modulus p^r for prime p and integer r. Plaintext polynomial $m \in \mathcal{R}_{p^r}$
- Suitable for computing on modular integers & finite fields

- Secret key generation: $s \leftarrow \chi_s$ for some distribution χ_s over $\mathcal{R}$
- $\mathrm{Enc}(m; p^r) = c = (b, a) = (-as + m + p^r e, a)$, with $a \leftarrow \mathcal{R}_q, e \leftarrow \chi_e$
- $\mathrm{Dec}(c) = \left[[b + as]_q\right]_{p^r}$

Plaintext Batching in BGV

- $\mathcal{R}_{p^r}$ is isomorphic to E^L for some Galois field (or ring) E, let $d = [E : \mathbb{Z}_{p^r}]$
- Each of the L positions is called a **slot**
- SIMD property: ciphertext additions & multiplications are slot-wise

Bootstrapping in FHE

- All currently known FHE schemes rely on noisy encryption, e.g., NTRU, LWE, RLWE, AGCD...
- The noise in ciphertexts accumulates during homomorphic computation
- A too large noise will lead to decryption failure

- Bootstrapping: reset the noise level by evaluating the decryption circuit homomorphically
  - Much more expensive than homomorphic arithmetic operations in CKKS/BFV/BGV
  - The performance bottleneck of all FHE schemes

BGV Bootstrapping

- General bootstrapping: each slot stores a finite field/ring element
- **Homomorphic Digit Removal** is the performance bottleneck

$$
\begin{aligned}
&\mathrm{Enc}(m; p^r) \\
&\quad\downarrow \text{Decryption Formula Simplification and Homomorphic Inner Product} \\
&\mathrm{Enc}(p^{e-r}m + \epsilon; p^e) \\
&\quad\downarrow \text{CoeffToSlot and Unpacking} \\
&\mathrm{Enc}([p^{e-r}m_{i\cdot n} + \epsilon_{i\cdot n}, \ldots, p^{e-r}m_{i\cdot n+n-1} + \epsilon_{i\cdot n+n-1}]; p^e) \quad \text{for } i = 0, 1, \ldots, d-1 \\
&\quad\downarrow \text{Homomorphic Digit Removal} \\
&\mathrm{Enc}([m_{i\cdot n}, \ldots, m_{i\cdot n+n-1}]; p^r) \qquad\qquad \text{for } i = 0, 1, \ldots, d-1 \\
&\quad\downarrow \text{Repacking and SlotToCoeff} \\
&\mathrm{Enc}(m; p^r)
\end{aligned}
$$

BGV Bootstrapping

- Thin bootstrapping [CH18]: each slot stores an integer
- **Homomorphic Digit Removal** is still the performance bottleneck

$$\mathrm{Enc}([m_0, \ldots, m_{n-1}]; p^r)$$
$$\downarrow \text{SlotToCoeff}$$
$$\mathrm{Enc}(m; p^r)$$
$$\downarrow \text{Decryption Formula Simplification and Homomorphic Inner Product}$$
$$\mathrm{Enc}(p^{e-r}m + \epsilon; p^e)$$
$$\downarrow \text{CoeffToSlot}$$
$$\mathrm{Enc}([p^{e-r}m_0 + \epsilon_0, \ldots, p^{e-r}m_{n-1} + \epsilon_{n-1}]; p^e)$$
$$\downarrow \text{Homomorphic Digit Removal}$$
$$\mathrm{Enc}([m_0, \ldots, m_{n-1}]; p^r)$$

Homomorphic Digit Removal

- Homomorphically extract the highest r p-ary digits (m_i) from e digits $(p^{e-r}m_i + \epsilon_i)$
- $w_{e-1}w_{e-2}\cdots w_{e-r}w_{e-r-1}\cdots w_0 \rightarrow w_{e-1}\cdots w_{e-r}$

- Basic operation is the digit extraction: obtain $\underbrace{0\cdots 0}_{e-1} w_0$ from $w_{e-1}\cdots w_0$
- HS15 & HS21: Compute a lifting polynomial of degree p for $e-1$ times
- CH18: Compute a digit extraction polynomial of degree $(p-1)(e-1)+1$
- GIKV23: Compute two polynomials of degrees $(p-1)(e'-1)+1$ and $\left\lceil \frac{e}{e'} \right\rceil p$
- Polynomial degree at least p -> Inefficient for large p (p is at most 257 in previous works)

[HS15] S. Halevi, V. Shoup: Bootstrapping for HElib. EUROCRYPT 2015
[HS21] S. Halevi, V. Shoup: Bootstrapping for HElib. J. Cryptol. (2021)
[CH18] H. Chen, K. Han: Homomorphic Lower Digits Removal and Improved FHE Bootstrapping. EUROCRYPT 2018
[GIKV23] R. Geelen, I. Iliashenko, J. Kang, F. Vercauteren: On Polynomial Functions Modulo pe and Faster Bootstrapping for Homomorphic Encryption. EUROCRYPT 2023

Faster BGV Bootstrapping using Null Polynomials

- Definition [GIKV23]: A polynomial $f \in \mathbb{Z}_{p^e}[X]$ is called a *null polynomial over* $S \subseteq \mathbb{Z}_{p^e}$ if
$$f(a) \equiv 0 \bmod p^e, \forall a \in S.$$
- First exploited by Geelen et al. to accelerate digit removal at Eurocrypt'23.
- Digit removal consists of homomorphic polynomial evaluations, e.g., computing $f(\cdot)$ on $\mathrm{Enc}(a; p^e)$
- If there exists a null polynomial g over the support of a, with $\deg(g) \leq \deg(f)$, we have
$$(f \bmod g)(a) \equiv f(a) \bmod p^e$$
- Lower polynomial degree => less time & level consumption for digit removal

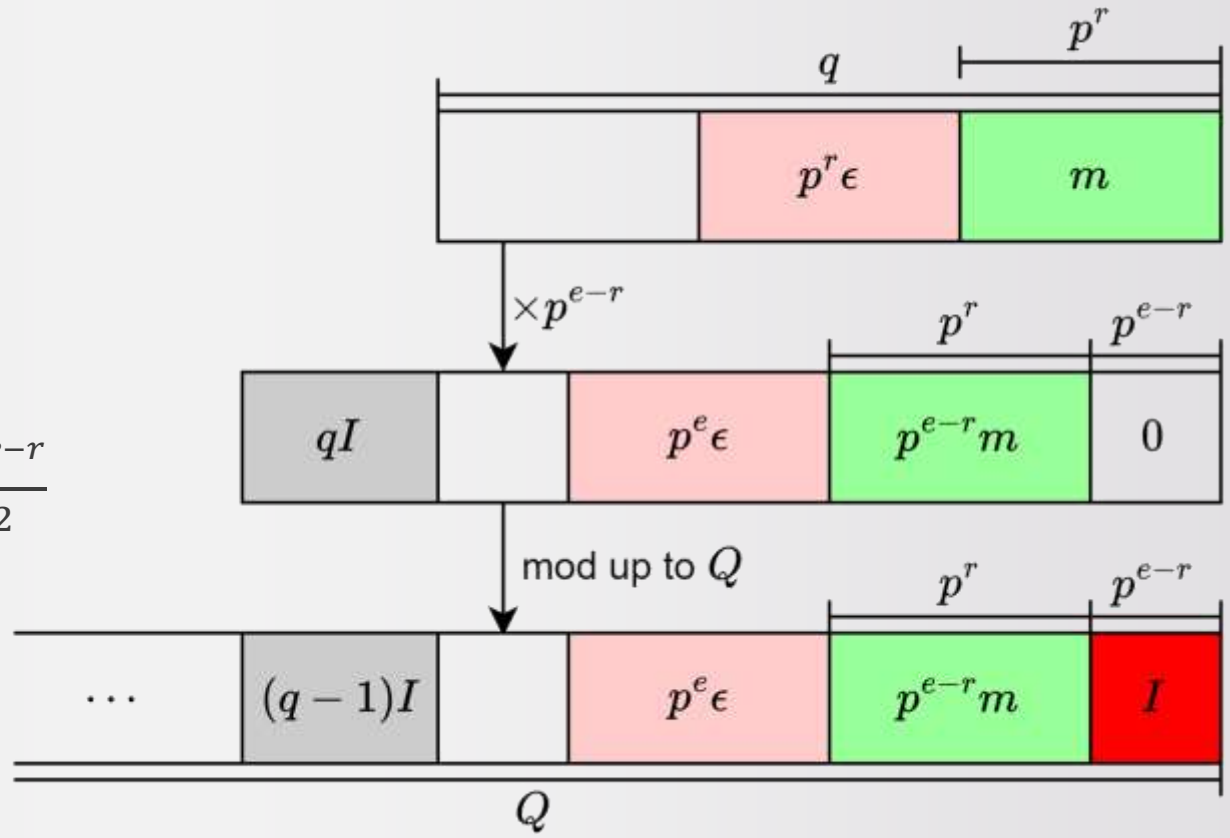
- Our target: find low-degree null polynomials

Accelerating Digit Removal: First Approach

- Core observation: $|\epsilon_i|$ can be made very small
- Consequence: there exists a low-degree null polynomial over $p^{e-r}m_i + \epsilon_i$

- First steps in BGV bootstrapping →

- Assume $q \equiv 1 \bmod p^e$ for simplicity
- Require $|p^{e-r}(m + p^r\epsilon)| < \frac{q}{2}$ and $|I| < \frac{p^{e-r}}{2}$
- $I = \epsilon$



The Bound on I

- $I = \left\lfloor \frac{[p^{e-r}b]_q + [p^{e-r}a]_q s}{q} \right\rceil$ has the size of a modulus-switching error
- Approximately, $\Pr\left[|I| > k\sqrt{\frac{h\phi(M)2^{\omega(M)}}{12M}}\right] < \phi(M) \cdot \mathrm{erfc}\left(\frac{k}{\sqrt{2}}\right)$ ▷ Heuristic by Halevi and Shoup

- For $k = 8, \omega(M) = 2, M < 2^{16}, \Pr\left[|I| > 4.7\sqrt{h}\right] < 2^{-33}$
- The bound on $|I|$ depends heavily on the Hamming weight h of s
- Solution: sparse secret key encapsulation by Bossuat et al. at ACNS'22
- $|I| \approx 23$ for our parameters

Low-degree Null Polynomials from Small Lower Digits

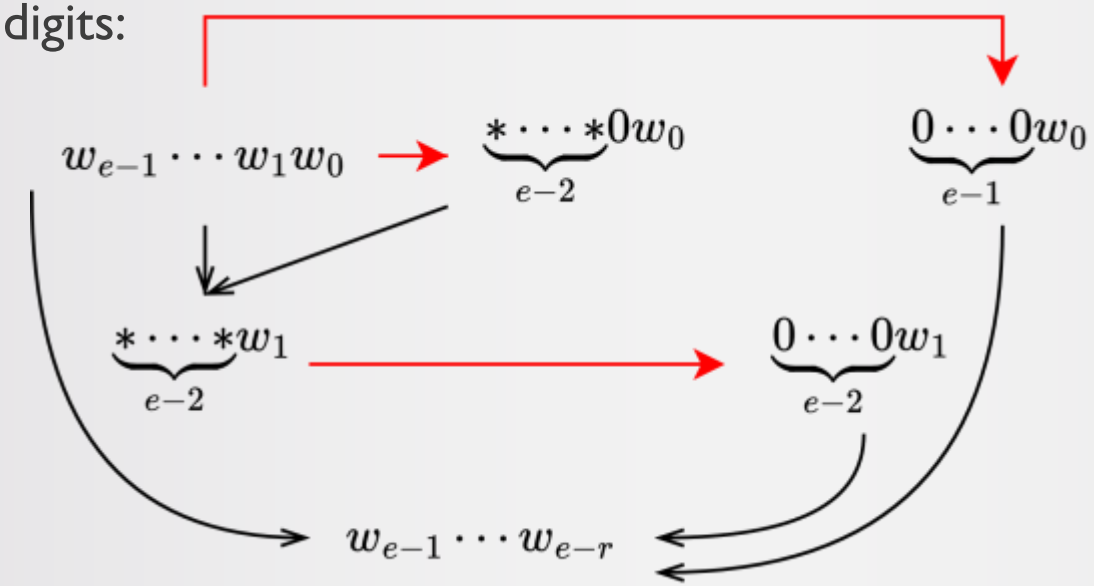
- Let $S = \{a \mid a = p^{e-r}m_i + I_i\}$ using the bound on I
- Let $B = |I|$, let k satisfy $e \leq k\left(e - r + v_p\left((2B)!\right) - \left\lfloor \log_p(2B) \right\rfloor\right) + v_p(k!)$. Let

$$\Lambda(X) = \prod_{j=0}^{k-1}\left(\prod_{i=-B}^{B}(X-i) - j \cdot p^{e-r+v_p\left((2B)!\right)-\left\lfloor \log_p(2B)\right\rfloor}\right).$$

- $\Lambda(X)$ is a null polynomial over S of degree at most $\left\lceil \frac{e}{e-r} \right\rceil (2B+1)$
- Much smaller than p when p is large. E.g., $p = 65537, r = 1, \deg\left(\Lambda(X)\right) < 100$

Applying to Digit Removal

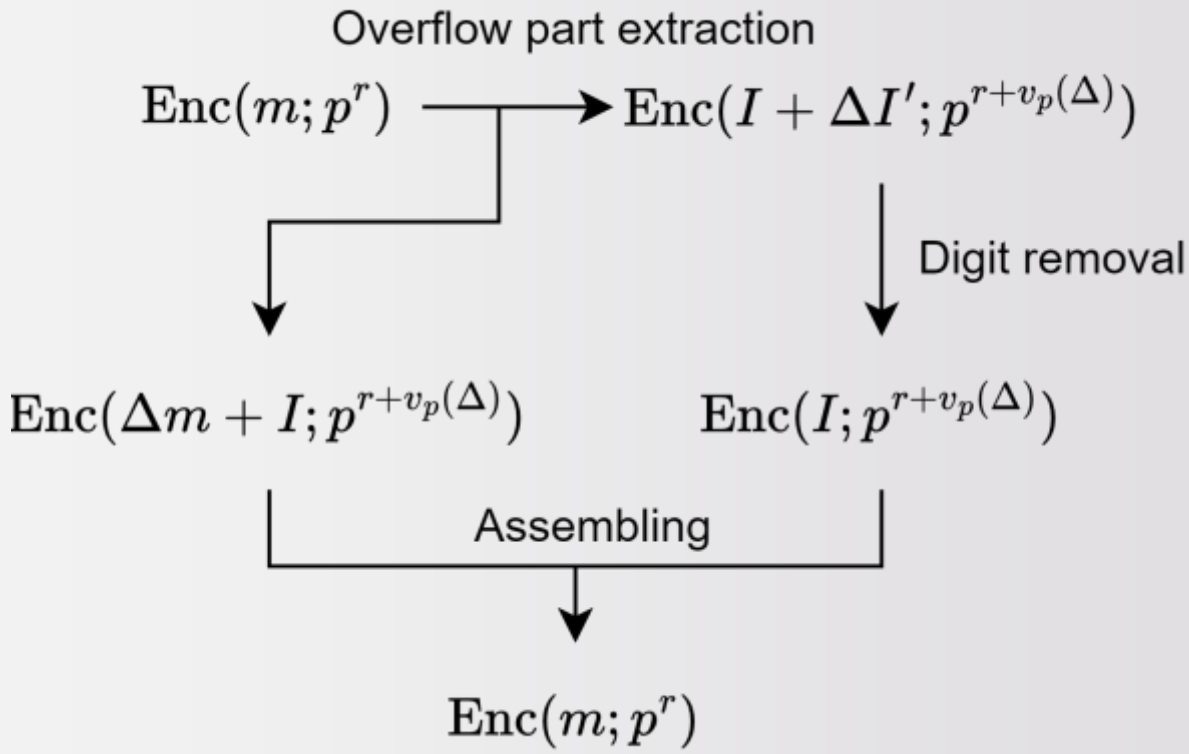
- For a coefficient of $p^{e-r}m + I$, let $w_{e-1} \cdots w_0$ be its p-ary representation
- If $|I|$ takes one digit: extracting $\underbrace{0 \cdots 0}_{e-1} w_0$ from $w_{e-1} \cdots w_0$ finishes the digit removal step
- If $|I|$ takes two digits:



- Support for $p \geq 11$
- More than two digits are not considered because p is too small

Accelerating Digit Removal: Second Approach

- Core observation: the input to digit removal can be made into $I + \Delta I'$ using the technique from KDE+24
- Both I and I' are bounded by B, independent of m
- Consequence: a null polynomial of degree $(2B+1)^2$ over the coefficients of $I + \Delta I'$
- Usually have higher degree than null polynomials in the first approach

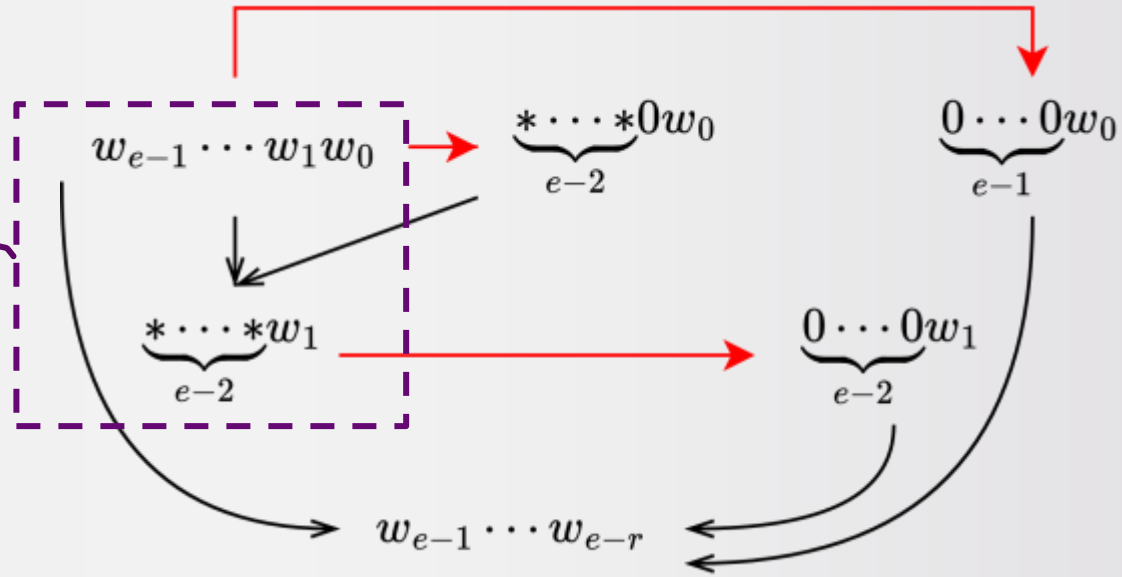


1. $\Delta = p^{e-r}: B < \frac{\Delta}{2}$
2. Δ coprime with p: $B < \frac{\Delta}{2}, |I + \Delta I'| < \frac{p}{2}$

KDE+24 Andrey Kim, Maxim Deryabin, Jieun Eom, Rakyong Choi, Yongwoo Lee, Whan Ghang, Donghoon Yoo: General Bootstrapping Approach for RLWE-based Homomorphic Encryption. IEEE Trans. Computers 73(1): 86-96 (2024)

Low-degree Interpolation Polynomial from Small Support Size

- Case 1: $\Delta = p^{e-r}$
- Support sizes are no more than $(2B+1)^2$
- $\Lambda_a(\mathrm{X}) = \prod_{b \in \mathrm{supp}(a)} (X - b)$



Low-degree Interpolation Polynomial from Small Support Size

- Case 2: Δ coprime with p
- We require $B \leq \frac{\Delta}{2}, B + \Delta B < \frac{p}{2}$
- There is an interpolation polynomial $f(X)$ of degree $(2B+1)^2 - 1$, satisfying

$$f\left(I_j + \Delta I'_j\right) = I_j \bmod p^r$$

  where $I_j, I'_j \in \mathbb{Z}_{p^r}$ are coefficients of I and I'.

- The plaintext modulus remains p^r throughout bootstrapping since $v_p(\Delta) = 0$

Parameter Sets

Table 4: The selected parameter sets are categorized as follows: Parameter sets that share the same Roman number have identical values of p, r, M. Type-A parameter sets utilize $\Delta = p^t$, while type-B parameter sets employ $\Delta = \Delta_0$ coprime to p. Other parameter sets use HElib's original bootstrapping (HS bootstrapping). Parameters that provide 80-bit security are enclosed in braces.

ID	p^r	M	d	$\log_2(Q)$	λ'	h	λ	$\log_2(\epsilon)$	B	Δ or $p^{e-e'}$	$\log_2(q_0 R)$
I	17^4	38309	24	1462	82.5	24(12)	136.0(81.3)	−34.3	/	17^2	63.8(64.5)
I-A						24(14)	133.1(89.8)		23(18)		70.3
II	127^2	56647	45	2253	82.6	22(12)	134.1(87.3)	−33.7	/	127	69.4(60.7)
II-A						22(12)	135.4(85.7)		22(17)		66.6
III	257^2	55427	28	2176	82.8	22(12)	135.5(87.4)	−33.8	/	257	64.2(64.7)
III-A						22(12)	133.4(85.5)		22(17)		70.6
IV	8191	45193	14	1803	82.3	22(12)	130.2(83.3)	−34.0	/	8191	66.3(66.8)
IV-A						24(12)	136.7(81.9)		23(17)		72.7
IV-B						22(12)	131.8(83.8)	−33.0	22(17)	45	62.8
V	65537	50731	18	2036	82.3	22(12)	130.6(83.0)	−33.9	/	65537	74.6(75.1)
V-A						24(12)	136.8(82.3)		23(17)		81.2
V-B						22(12)	132.8(84.8)	−32.9	23(17)	47	67.1

Degree of Polynomials, HElib and Ours

Parameter Set	HElib	Ours-A	Ours-B
I	17, 81; 65	17, 81; 14 ($\approx 1/4.5$)	NA
II	253	134 ($\approx 1/1.9$)	NA
III	513	134 ($\approx 1/3.8$)	NA
IV	8191	93 ($\approx 1/88$)	2024 ($\approx 1/4$)
V	65536	93 ($\approx 1/700$)	2208 ($\approx 1/32$)

Experimental Results

Table 6: Benchmark results of general bootstrapping with 128 bits of security for s. The parameter sets without any suffix use the HS bootstrapping provided by HElib. The parameter sets suffixed with '-A' use $\Delta = p^t$, where both the global and local null polynomial optimizations are available. Those suffixed with '-B' use $\Delta = \Delta_0$, where the digit removal is performed using the lifted interpolation polynomial (i.e., only the global null polynomial optimization is available).

Parameter Set ID		I	I-A	II	II-A	III	III-A	IV	IV-A	IV-B	V	V-A	V-B
Capacity (bits)	Initial	1003	1019	1573	1594	1542	1558	1253	1266	1287	1415	1431	1457
	Linear map	207	241	194	219	208	241	199	243	158	225	277	171
	Digit extract	446	341	298	293	400	316	559	288	329	812	331	389
	Remaining	344	434	1070	1080	927	998	484	732	791	363	820	889
Time (sec)	Linear map	93	93	285	277	319	319	113	115	113	115	112	115
	Digit extract	953	568	3176	2148	3010	1224	4535	363	2095	46088	574	3471
	Total	1046	662	3462	2427	3330	1545	4648	479	2209	46203	688	3589
Throughput (bps)		0.329	0.656	0.309	0.445	0.278	0.646	0.104	1.527	0.358	0.008	1.191	0.248
Speedup		1x	2.00x	1x	1.44x	1x	2.32x	1x	14.7x	3.44x	1x	151x	31.5x

Summary of Idea

- The sparsity of I in $p^{e-r}m + I$, the input to homomorphic digit removal, can greatly accelerate the bootstrapping process by lowering the polynomial degree.
  - There exists null polynomials over $p^{e-r}m + I$ of degree roughly $2|I|\frac{e}{e-r}$.
- The idea has been used in CKKS bootstrapping, where computing m from $q_0 I + m$ also benefits from the property that $|I|$ is small.
  - $|I|$ allows to approximate the mod function on a smaller range, thus with polynomials of lower degrees.
- I has the same expression in BGV and CKKS (as a function of the Hamming weight). Both schemes benefit from a sparse bootstrapping key.

Thank you for your attention

- Q&A