Revisiting Homomorphic Encryption Schemes for Finite Fields

Andrey Kim[1], Yuriy Polyakov[2], **Vincent Zucca**[3]

[1]Samsung Advanced Institute of Technology, Suwon, Republic of Korea
[2]Duality Technologies, Newark, USA
[3]LIRMM-DALI, Université de Perpignan Via Domitia, France

Asiacrypt 2021

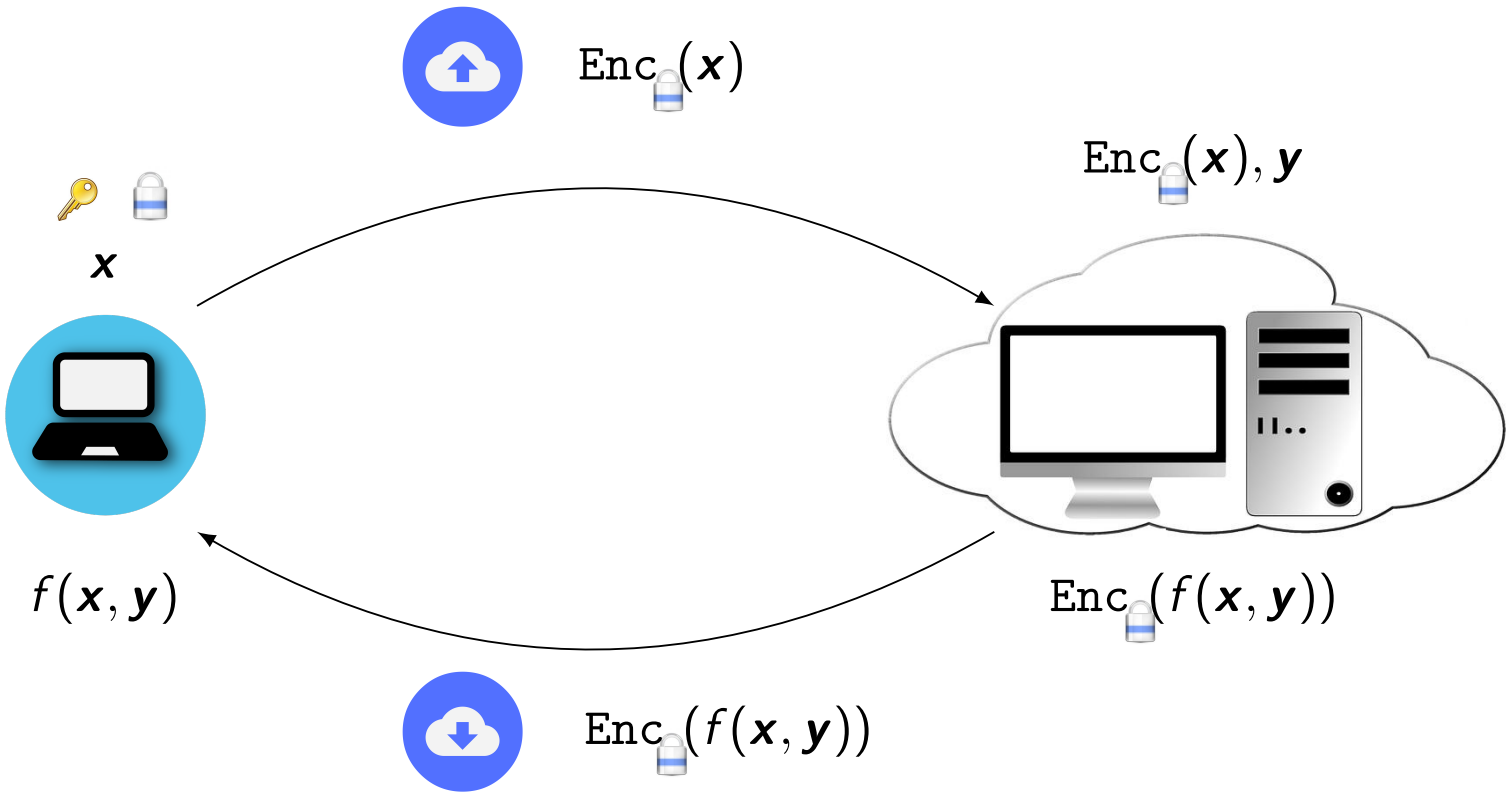
SAMSUNG ADVANCED INSTITUTE OF TECHNOLOGY

What is Homomorphic Encryption (HE)?

HE allows to compute over encrypted data without the decryption key

Applications:

- Private search queries;
- Secure multi-party computations;
- Delegation of computations over sensitive data.



Homomorphic Encryption from Ring-LWE

- Consider the ring $\mathcal{R}_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ with $N = 2^d$
- Informally, the Ring-LWE problem states that if
  - $\boldsymbol{s} \xleftarrow{\$} \chi_{\texttt{key}}$ the secret (uniform in $\{-1, 0, 1\}^n$)
  - $\boldsymbol{a}_i \xleftarrow{\$} \mathcal{U}_Q$ coeff. uniform $[\cdot]_Q$ (in $[-Q/2, Q/2) \cap \mathbb{Z}$)
  - $\boldsymbol{e}_i \xleftarrow{\$} \chi_{\texttt{err}}$ coeff. drawn from centred distribution with std dev $\sigma_{\texttt{err}}$

  then $[\boldsymbol{a}_i \cdot \boldsymbol{s} + \boldsymbol{e}_i]_Q \in \mathcal{R}_Q$ looks uniform over $\mathcal{R}_Q$ even if we know $\boldsymbol{a}_i$.
- We can thus use $([\boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_Q, \boldsymbol{a})$ to mask a message $\boldsymbol{m} \in \mathcal{R}$

$$\texttt{ct} = (\boldsymbol{c}_0, \boldsymbol{c}_1) = ([\boldsymbol{m} + \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e}]_Q, -\boldsymbol{a}) \in \mathcal{R}_Q^2$$

- Knowing $\boldsymbol{s}$, one can recover a "noisy" version of the message $\boldsymbol{m}$

$$\texttt{ct}(\boldsymbol{s}) = \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot \boldsymbol{s} = \boldsymbol{m} + \mathbf{e} \bmod Q$$

Homomorphic Encryption from Ring-LWE

$$\mathtt{ct}(\boldsymbol{s}) = \boldsymbol{m} + \mathbf{e} \bmod Q$$

- To get rid of the noise $\mathbf{e}$, one can encode a bounded message $[\boldsymbol{m}]_t \in \mathcal{R}_t$ ($t \ll Q$) in the LSB or the MSB of the ciphertext
  - $\mathtt{ct}(\boldsymbol{s}) = [\boldsymbol{m}]_t + t\mathbf{e} \bmod Q \rightarrow \mathtt{BGV}$

  - $\mathtt{ct}(\boldsymbol{s}) = \lfloor Q/t \rfloor [\boldsymbol{m}]_t + \mathbf{e} \bmod Q \rightarrow \mathtt{BFV}$

- Works only as long as $\|\mathbf{e}\|_\infty \leq B_{\mathtt{dec}} \approx \frac{Q}{2t}$

Homomorphic Encryption from Ring-LWE

- Schemes based on Ring-LWE are naturally homomorphic ($+$ and $\times$)
  - Addition (BGV)

$$\begin{aligned}\mathtt{ct}(\boldsymbol{s}) + \mathtt{ct}'(\boldsymbol{s}) &= [\boldsymbol{m}]_t + [\boldsymbol{m}']_t + t(\boldsymbol{e} + \boldsymbol{e}') \\ &= [\boldsymbol{m} + \boldsymbol{m}']_t + t(\mathbf{e} + \mathbf{e}' + \mathbf{u}) \bmod Q\end{aligned}$$

$$\rightarrow \mathtt{ct}_{\mathsf{add}} = ([\boldsymbol{c}_0 + \boldsymbol{c}'_0]_Q, [\boldsymbol{c}_1 + \boldsymbol{c}'_1]_Q) \in \mathcal{R}_Q^2$$

  - Multiplication (BGV)

$$\begin{aligned}\mathtt{ct}(\boldsymbol{s}) \cdot \mathtt{ct}'(\boldsymbol{s}) &= [\boldsymbol{m}]_t \cdot [\boldsymbol{m}']_t + t(t\boldsymbol{e} \cdot \boldsymbol{e}' + [\boldsymbol{m}]_t \cdot \boldsymbol{e}' + [\boldsymbol{m}']_t \cdot \boldsymbol{e}) \\ &= [\boldsymbol{m} \cdot \boldsymbol{m}']_t + t(\mathbf{te} \cdot \mathbf{e}' + \mathbf{[m]_t} \cdot \mathbf{e}' + \mathbf{[m']_t} \cdot \mathbf{e} + \mathbf{r}) \bmod Q\end{aligned}$$

$$\rightarrow \mathtt{ct}_{\mathsf{mult}} = ([\boldsymbol{c}_0 \cdot \boldsymbol{c}'_0]_Q, [\boldsymbol{c}_0 \cdot \boldsymbol{c}'_1 + \boldsymbol{c}'_0 \cdot \boldsymbol{c}_1]_Q, [\boldsymbol{c}_1 \cdot \boldsymbol{c}'_1]_Q) \in \mathcal{R}_Q^3$$

- The noise grows with homomorphic operations
  - multiplications introduce more noise than additions
  - choose Q (and N) accordingly

RNS implementation of HE

- In practice the modulus Q can be very large (hundreds of bits)
- For efficiency reasons Q is usually chosen as $Q = q_1 \cdots q_k$
  - q_i small distinct prime moduli ($\log_2 q_i < 64$)
- CRT states we have the following ring isomorphism

$$\mathcal{R}_Q \simeq \mathcal{R}_{q_1} \times \cdots \times \mathcal{R}_{q_k}$$

  - $+$ and $\times$ of $\mathcal{R}_Q$ elements can be done in parallel over the $\mathcal{R}_{q_i}$s
  - avoid multiprecision arithmetic and carry propagation
  - $\{q_1, \cdots, q_k\}$ is called the RNS basis

Contributions of this work

- BGV and BFV are theoretically equivalent, yet significant differences remain in practice
  - more efficient homomorphic multiplication for BGV
    - ★ BGV is faster than BFV
  - using BGV efficiently requires to handle the noise dynamically
    - ★ BFV is simpler to implement and to use than BGV
  - better noise growth for BGV with large plaintext modulus [CLP20]
    - ★ more homomorphic operations with BGV on the same parameters
- This work focuses on closing the gap between the two schemes
  - modify BFV encoding $\rightarrow$ noise growth somewhat equivalent to BGV
  - reduce the algorithmic complexity of BFV homomorphic multiplication
  - develop a more usable variant of BGV
- This work also introduces several novel optimizations for BFV
  - separation of tensoring and scaling in BFV homomorphic multiplication
  - improvement of the RNS decryption procedure of [HPS19]

Comparison of BGV and BFV noise growth

- BGV ciphertexts can be decrypted by computing

$$\mathsf{ct}(\boldsymbol{s}) = [\boldsymbol{m}]_t + t\mathbf{v} \bmod Q$$

$$\|\mathbf{v}\|_\infty < \frac{Q}{2t} - \frac{1}{2}$$

- BFV decryption procedure is slightly more complicated

$$\mathsf{ct}(\boldsymbol{s}) = \lfloor Q/t \rfloor\, [\boldsymbol{m}]_t + \mathbf{v} \bmod Q$$

$$\left\lfloor \frac{t}{Q}[\mathsf{ct}(\boldsymbol{s})]_Q \right\rceil = [\boldsymbol{m}]_t + \left\lfloor \frac{t\mathbf{v} - r_t(Q)[\boldsymbol{m}]_t}{Q} \right\rceil \bmod t$$

$$\|\mathbf{v}\|_\infty < \frac{Q}{2t} - \frac{r_t(Q)}{2}$$

  - $r_t(Q) \in [0, t)$ comes from the difference between t/Q and $\lfloor Q/t \rfloor^{-1}$

$$Q = t\lfloor Q/t \rfloor + r_t(Q) \implies \frac{t}{Q}\lfloor Q/t \rfloor = 1 - \frac{r_t(Q)}{Q}$$

Comparison of BGV and BFV noise growth

- In BFV, the size of the noise after an homomorphic multiplication can be approximated by

$$\delta_{\mathcal{R}}^2 t \left(V_{\text{input}} + \frac{r_t(Q)}{2} \right)$$

  - the $r_t(Q)$ term comes from the scaling by t/Q required in BFV homomorphic multiplication.

- What is the dominant term between V_{input} and $r_t(Q)/2$?
  - a fresh ciphertext has its noise bounded by $V_{\text{init}} < 77\sqrt{N} < 2^{15}$
  - $r_t(Q) > 2^{15} \implies$ larger noise growth for the first multiplication
    - ★ $t = 2^{32}$ and $r_t(Q)/2 \approx 2^{31} \rightarrow$ 16 bits of difference
    - ★ $t = 2^{60}$ and $r_t(Q)/2 \approx 2^{59} \rightarrow$ 44 bits of difference
  - after first multiplication V_{input} becomes dominant $\rightarrow$ no further growth
  - matches the experiments of [CLP20]

Comparison of BGV and BFV noise growth

- $r_t(Q)$ impacts both the decryption and homomorphic multiplication
- One could choose the moduli q_i such that $r_t(Q)$ is small
  - more a patch than a real solution
  - restrict further the choice of the moduli q_i
- Change the message encoding from $\lfloor Q/t \rfloor [\boldsymbol{m}]_t$ to $\lfloor Q/t [\boldsymbol{m}]_t \rceil$

$$\frac{t}{Q}\left\lfloor \frac{Q[\boldsymbol{m}]_t}{t} \right\rceil = [\boldsymbol{m}]_t + \frac{t}{Q}\varepsilon$$

  - decoding error reduced from $\frac{r_t(Q)t}{2Q}$ to $\frac{t}{2Q}$
  - same decryption and noise growth bounds as for BGV
- When t and Q are coprime $\lfloor Q/t [\boldsymbol{m}]_t \rceil$ can be computed in RNS as

$$\left\lfloor \frac{Q[\boldsymbol{m}]_t}{t} \right\rceil = \frac{Q[\boldsymbol{m}]_t - [Q\boldsymbol{m}]_t}{t} = -\frac{[Q\boldsymbol{m}]_t}{t} \bmod Q$$

Improving BFV homomorphic multiplication

$$\mathtt{ct}(\boldsymbol{s}) = \frac{Q}{t}[\boldsymbol{m}]_t + \boldsymbol{v} + Q\boldsymbol{k}$$

- In BFV the tensoring must be done without reduction modulo Q

$$\mathtt{ct}(\boldsymbol{s}) \cdot \mathtt{ct}'(\boldsymbol{s}) = \frac{Q^2}{t^2}[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \frac{Q}{t}\boldsymbol{v}_{\text{tensor}} + \frac{Q^2}{t}\boldsymbol{k}_{\text{tensor}}$$

$$\text{with } \boldsymbol{k}_{\text{tensor}} = [\boldsymbol{m}]_t \cdot \boldsymbol{k}' + [\boldsymbol{m}']_t \cdot \boldsymbol{k} + \mathbf{tk} \cdot \mathbf{k'} + \boldsymbol{r}$$

- Use a second RNS basis $P = p_1 \cdots p_{k'}$ so that $\frac{t}{Q}\mathtt{ct}(\boldsymbol{s}) \cdot \mathtt{ct}'(\boldsymbol{s})$ does not wrap around modulo P

$$\frac{t}{Q}\mathtt{ct}(\boldsymbol{s}) \cdot \mathtt{ct}'(\boldsymbol{s}) = \frac{Q}{t}[\boldsymbol{m} \cdot \boldsymbol{m}']_t + \boldsymbol{v}_{\text{tensor}} + Q\boldsymbol{k}_{\text{tensor}}$$

  - the dominant term after the rounding is $\mathbf{tQk} \cdot \mathbf{k'}$ mod P
  - $P > t\delta_{\mathcal{R}}^3 Q/4 \rightarrow$ in practice taking $k' = k + 1$ is enough

Improving BFV homomorphic multiplication

- The dominant term after the rounding is $\mathbf{tQk} \cdot \mathbf{k}'$ mod P
  - transform the multiple of Q to a multiple of P to cancel this term

- Switch the modulus from Q to P on one of the two ciphertexts

$$\mathrm{ct}(\boldsymbol{s}) = \frac{P}{t}[\boldsymbol{m}]_t + \frac{P}{Q}\boldsymbol{v} + P\boldsymbol{k} + \varepsilon$$

- Perform the tensoring modulo PQ as usual

$$\mathrm{ct}(\boldsymbol{s})\cdot\mathrm{ct}'(\boldsymbol{s}) = \frac{PQ}{t^2}[\boldsymbol{m}\cdot\boldsymbol{m}']_t + \frac{P}{t}\boldsymbol{v}_{\text{tensor}} + \frac{QP}{t}\boldsymbol{k}_{\text{new-tensor}} + \varepsilon\cdot\mathrm{ct}'(\boldsymbol{s}) \bmod PQ$$

$$\text{with } \boldsymbol{k}_{\text{new-tensor}} = [\boldsymbol{m}]_t \cdot \boldsymbol{k}' + [\boldsymbol{m}']_t \cdot \boldsymbol{k} + \boldsymbol{r}$$

  - the term $QP\boldsymbol{k} \cdot \boldsymbol{k}'$ vanishes modulo PQ

- Scale down by t/P instead of t/Q
  - the scaling can be done directly in base Q

Improving BFV homomorphic multiplication

- The rounding error ε caused by the modulus switching from Q to P adds to the noise
- The size of this error can be controlled with P
  - taking $P \approx Q$ makes this noise negligible
  - one can take $k' = k$ instead of $k' = k + 1$
- It is possible to improve further the performance of the homomorphic multiplication by considering a "leveled" approach
  - scale down both ciphertexts from Q to Q' (with $Q' \mid Q$)
  - perform the multiplication internally modulo $P'Q'$ instead of PQ
  - scale up from Q' to Q to get back ciphertexts modulo Q
- Requires to have a lower bound on the noise of a ciphertext to scale down appropriately
  - Q' too small $\implies$ noise added by modulus-switching dominates
  - problematic somewhat similar to BGV

More usable BGV variant

- Unlike for BFV the noise is not scaled down automatically during multiplication for BGV
  - the noise must be reduced "manually" to prevent a quadratic blow-up
- Modulus switching is fundamental for BGV
  - requires a precise noise estimation to scale down appropriately
  - we loose one "level" after each multiplication
- HElib includes a dynamic noise estimation mechanism to keep track of the size of the noise of each ciphertext
  - the noise is scaled down before every multiplication to a minimum level
  - requires an additional set of "small" primes to scale down by the right ammount
- We propose an alternative approach without any dynamic estimation
  - the user needs to indicate the maximum number of additions/rotations he wants to perform between two multiplications beforehand
  - the moduli are chosen during key generation accordingly
  - the "constant" noise level is set to twice the modulus switching noise.

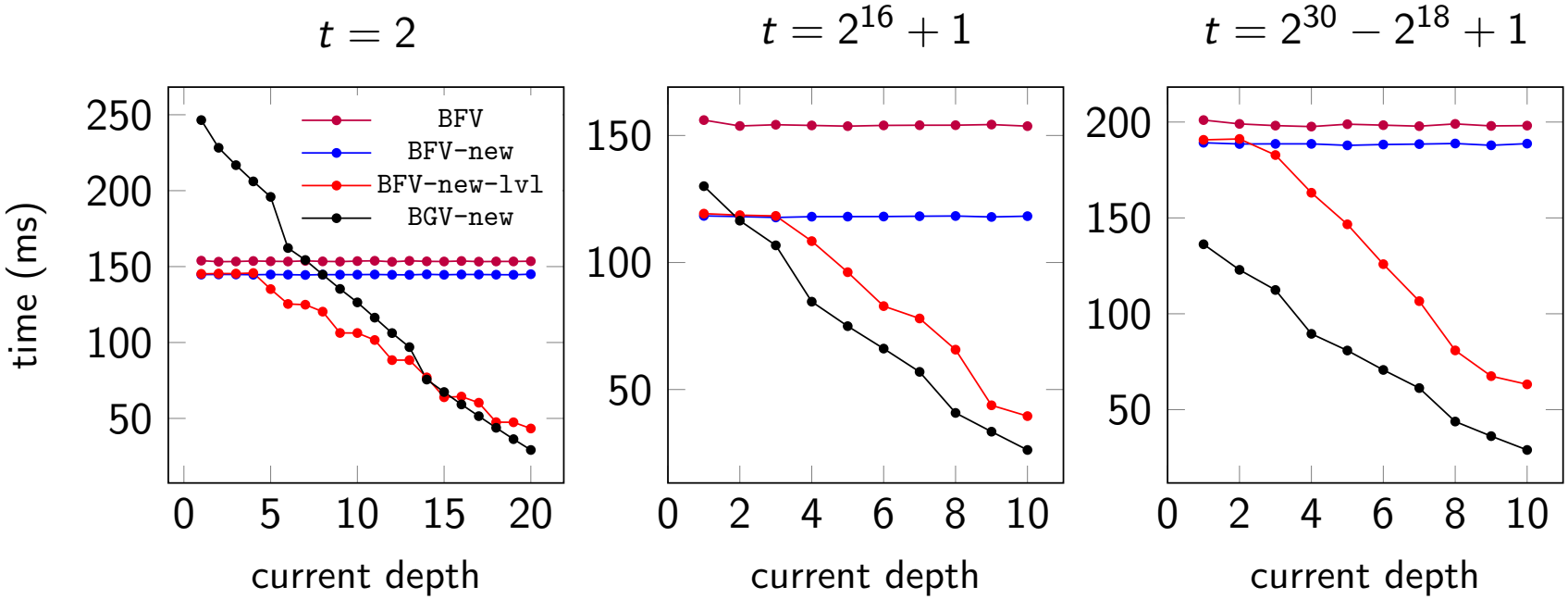
Comparison of BGV and BFV

- Both schemes have the same decryption bound $B_{\mathsf{dec}} = Q/2t - 1/2$
- The noise after multiplication in BFV is dominated by $\delta_{\mathcal{R}}^2 t V_{\mathsf{input}}$
- The noise after multiplication in BGV is dominated by $2\delta_{\mathcal{R}}^2 t V_{\mathsf{input}}$
  - at each level the down-scaled noise matches the noise added by modulus-switching $\rightarrow$ factor 2
- In BGV the noise is kept at a small size ($\approx \delta_{\mathcal{R}}$) throughout the computations
  - the noise added by key switching $\approx 3k\delta_{\mathcal{R}}\sigma_{\mathsf{err}}$ can dominate the multiplication noise when the plaintext modulus is small (e.g. $t = 2$).
  - small advantages for BFV for small plaintext modulus

Performance of BGV and BFV

- We have incorporated our variants of BGV and BFV in the PALISADE C++ library
  - the last optimizations will be incorporated to the next release.
- First time that BGV and BFV are implemented in the same library
  - better comparison
- Experiments were run on a commodity desktop computer
  - Intel(R) Core(TM) i7-9700 CPU @ 3.00 GHz
  - 64 GB of RAM
  - Ubuntu 18.04 LTS
  - single-threaded mode
- Experiments were run using hybrid key-switching with 3 digits

Performance of BGV and BFV



Homomorphic multiplication runtimes at various depths with N set to 2^{15}.

- BGV is slower at first depths for small and medium plaintext modulus
  - For BGV, the size of the moduli q_i depends on t
    - ★ t small $\Longrightarrow$ the number of moduli k increases
    - ★ the number of NTTs to compute is proportional to k
    - ★ experiments were run in a single-threaded mode
  - Possibility to trunk several moduli on a same machine word to gain in performance ($\times 2 - \times 3$)

Performance of BGV and BFV

	Original BFV				Our BFV						Our BGV			
	params		BFV		params		BFV-new		BFV-new-lvl		params		BGV-new	
ℓ	$\log N$	$\log Q$	$\log e$	time(s)	$\log N$	$\log Q$	$\log e$	time(s)	$\log e$	time(s)	$\log N$	$\log Q$	$\log e$	time(s)
1	13	62	44	0.01	13	59	36	0.004	35	0.004	13	58	34	0.005
2	13	94	66	0.03	13	90	63	0.025	63	0.025	13	91	67	0.02
3	14	129	103	0.21	14	123	95	0.19	96	0.18	13	124	100	0.063
4	14	159	132	0.45	14	156	125	0.4	125	0.39	13	157	133	0.17
5	14	192	161	1.2	14	188	155	1.07	155	1.04	14	196	171	0.8
6	14	224	189	2.44	14	220	184	2.18	184	2.13	14	230	205	2.03
7	14	255	220	6.51	14	250	214	5.98	214	5.73	14	264	239	4.86

Comparison of noise growth and runtimes of BGV and BFV variants for a benchmark computation $\prod_{i=1}^{2^\ell} x_i$. $t = 2^{16} + 1$, and $\lambda \geq 128$ bits.

- slightly faster noise growth for BGV (as expected)
- BGV has only a minor speed-up over BFV (no moduli trunking)

Performance of BGV and BFV

	Original BFV				Our BFV						Our BGV			
	params		BFV		params		BFV-new		BFV-new-lvl		params		BGV-new	
ℓ	$\log N$	$\log Q$	$\log e$	time(s)	$\log N$	$\log Q$	$\log e$	time(s)	$\log e$	time(s)	$\log N$	$\log Q$	$\log e$	time(s)
2	13	68	40	0.01	13	64	35	0.009	36	0.009 s	13	68	38	0.007
4	13	100	76	0.03	13	96	67	0.026	67	0.025 s	13	107	74	0.024
8	14	135	107	0.22	14	129	100	0.19	100	0.18 s	13	148	116	0.061
16	14	168	138	0.46	14	162	130	0.4	130	0.33 s	14	197	163	0.28
32	14	200	167	1.22	14	196	161	1.1	161	0.78 s	14	244	208	0.61
48	14	232	198	1.85	14	228	191	1.66	190	1.22 s	14	286	251	1.07
64	14	232	199	2.48	14	228	191	2.22	191	1.54 s	14	293	256	1.27

Comparison of noise growth and runtimes of BGV and BFV variants for a benchmark computation $\prod_{i=0}^{\ell} a_i x^i$: $|a_i| < 16$. $t = 2^{16} + 1$, and $\lambda \geq 128$ bits.

- siginificant higher noise for BGV than for BFV
  - the moduli q_i are chosen with extra-room for the additions
- again BGV has a minor speed-up over all the variants of BFV

Conclusion

- Our BFV variants have somewhat better noise growth than BGV
  - previous results suggested the opposite for large t
- For small t, BFV will be faster than BGV in single-threaded mode if the moduli are not trunked together
- We reduced the complexity of the homomorphic multiplication for BFV
  - the extra RNS basis now requires k, instead of $k+1$, moduli
  - we proposed a leveled variant of BFV multiplication
- We have implemented a variant of BGV which does not require any dynamic noise estimation
  - easier to use and to implement
  - performance cost since the size of the moduli is chosen conservatively