RSA Conference 2019

San Francisco | March 4 – 8 | Moscone Center

BETTER.

SESSION ID: CRYP-T08

An Improved RNS Variant of the BFV Homomorphic Encryption Scheme

Yuriy Polyakov

Associate Research Professor
New Jersey Institute of Technology

Joint work with Shai Halevi (IBM) and Victor Shoup (NYU)

RSA Conference 2019

Introduction to Homomorphic Encryption

Homomorphic Encryption

- Homomorphic Encryption (HE): A non-interactive secure computing approach to perform computations over encrypted sensitive data without ever decrypting them.
- Enables outsourcing of data storage/processing to a public cloud without compromising data privacy.
- HE schemes provide efficient instantiations of post-quantum public-key and symmetric-key encryption schemes.
- Homomorphic encryption can be viewed as a generalization of public key encryption.

HE vs Other Secure Computing Approaches

	HE	MPC	SGX
Performance	Compute-bound	Network-bound	
Privacy	Encryption	Encryption / Non-collusion	Trusted Hardware
Non-interactive	✔	✗	✔
Cryptographic security	✔	✔	✗ (known attacks)

Hybrid approaches are possible

Applications of Homomorphic Encryption

Domain	Genomics	Health	National Security	Education	Social Security	Business Analytics	Cloud
Sample Topics	*GWAS*	*billing and reporting*	*smart grid*	*school dropouts*	*credit history*	*prediction*	*storage, sharing*
Data Owner	*medical institutions*	*clinics and hospitals*	*nodes and network*	*schools, welfare*	*government*	*business owners*	*clients*
Why HE?	*HIPAA*	*cyber insurance*	*privacy*	*FERPA*	*cyber crimes*	*data are valuable*	*untrusted server*
Who pays?	*health insurance*	*hospital*	*energy company*	*DoE*	*government*	*business owners*	*clients*

Key Players in the HE Market

- HE is already practical for many applications, and is being commercialized
- Key players
  - Microsoft (SEAL library)
  - IBM (HELib library)
  - Duality Technologies (PALISADE library)

Key Concepts on Popular HE Schemes

- All popular schemes are based on large-degree (>1000) polynomials with integer coefficients.
- Integer coefficients are typically large and require multiprecision arithmetic (larger than 32 or 64 bits on typical systems).
- Popular schemes working with large-integer coefficients:
  - Brakerski-Gentry-Vaikuntanathan (BGV): fastest for exact number arithmetic
  - Brakerski/Fan-Vercauteren (BFV): most usable for exact number arithmetic
  - Cheon-Kim-Kim-Song (CKKS): ideal for approximate number arithmetic

PALISADE Lattice Cryptography Library

- Project-based Development since 2014
  - Funded by DARPA, IARPA, Sloan Foundation, NSA, and Simons Foundation
- Key Implementation Partners and Collaborators
  - Academia: MIT, UCSD, WPI, NUS, Sabanci U
  - Industry: Raytheon (BBN), Duality Technologies, IBM Research, Lucent, Vencore Labs, Galois, Two Six Labs
- BSD 2-clause license & Cross-Platform Support
- Implements HE schemes (BGV, BFV, etc.), proxy re-encryption, digital signatures, identity-based encryption, attribute-based encryption, etc.

Our Optimized Variant of the BFV Scheme

Why RNS is important?

- Benefits of Residue Number System (RNS) or Chinese Remainder Theorem (CRT) representation of polynomial coefficients
  - RNS works with native (machine-word size) integers: faster (up to 10x) and simpler than multi-precision integer arithmetic
  - Runtime scales (quasi)linearly with integer size
  - RNS dramatically improves memory locality
  - Computations are easily parallelizable
  - RNS supports efficient GPU/FPGA hardware implementations

Prior Work

- Double-CRT variant of BGV [GHS12]
- RNS variants of LTV (NTRU) scheme [CR14, DHS16], later implemented using FPGA and GPU
- Full RNS variant of BFV [BEHZ16]
  - Performs all operations in RNS
  - Uses sophisticated scaling and CRT extension techniques
  - Introduces auxiliary parameters (not present in BFV) and extra noise (which can be significant)
  - Normalized performance is about 2x slower than our variant

Challenges of Scale-Invariant Schemes (BFV)

- Decryption invariant

$$[\langle \mathbf{sk}, \mathbf{ct} \rangle]_q = m \cdot q/t + e, \text{ for a small noise term } |e| \ll q/t$$

- Scaling in decryption

$$m := \left[\left\lceil \frac{t}{q} \cdot [\langle \mathbf{sk}, \mathbf{ct} \rangle]_q \right\rfloor \right]_t$$

- Scaling in homomorphic multiplication (tensor product without modular reduction)

$$\mathbf{ct}^* := \left[\lceil t/q \cdot \mathbf{ct}_1 \otimes \mathbf{ct}_2 \rfloor \right]_q$$

- Ciphertext digit decomposition in key switching (relinearization)

Our Approach to CRT Basis Extension and Scaling Operations

- Big modulus is a smooth integer $q = \prod_i q_i$, where q_i are same-size, pair-wise coprime, single-precision integers (typically of size 30-60 bits)
- Use CRT reconstructions:

$$x = \Big(\sum_{i=1}^{k} \underbrace{[x_i \cdot \tilde{q}_i]_{q_i} \cdot q_i^*}_{\in \mathbb{Z}_q}\Big) - \upsilon \cdot q \text{ for some } \upsilon \in \mathbb{Z},$$

$$x = \Big(\sum_{i=1}^{k} \underbrace{x_i \cdot \tilde{q}_i \cdot q_i^*}_{\in [-\frac{q_i q}{4}, \frac{q_i q}{4})}\Big) - \upsilon' \cdot q \text{ for some } \upsilon' \in \mathbb{Z}.$$

$$q_i^* = q/q_i \in \mathbb{Z} \ \text{ and } \ \tilde{q}_i = {q_i^*}^{-1} \pmod{q_i} \in \mathbb{Z}_{q_i}$$

Our Approach to CRT Basis Extension

- Extend to modulus *p*

$$[x]_p = \left[\left(\sum_{i=1}^{k}[x_i \cdot \tilde{q}_i]_{q_i} \cdot q_i^*\right) - \upsilon \cdot q\right]_p$$

- Estimate υ (using floating-point arithmetic)

$$\upsilon = \left\lfloor \left(\sum_{i=1}^{k}[x_i \cdot \tilde{q}_i]_{q_i} \cdot q_i^*\right)/q \right\rfloor = \left\lfloor \sum_{i=1}^{k}[x_i \cdot \tilde{q}_i]_{q_i} \cdot \frac{q_i^*}{q} \right\rfloor = \left\lfloor \sum_{i=1}^{k} \frac{[x_i \cdot \tilde{q}_i]_{q_i}}{q_i} \right\rfloor$$

- Compute

$$[x]_p = \left[\left(\sum_{i=1}^{k} y_i \cdot [q_i^*]_p\right) - \upsilon \cdot [q]_p\right]_p$$

$$\text{where } y_i := [x_i \cdot \tilde{q}_i]_{q_i} \text{ and } \upsilon = \left\lfloor \sum_{i=1}^{k} \frac{y_i}{q_i} \right\rfloor$$

Our Approach to Scaling

$$y := \left\lceil \frac{t}{q} \cdot x \right\rfloor = \left\lceil \left(\sum_{i=1}^{k} x_i \cdot \tilde{q}_i \cdot q_i^* \cdot \frac{t}{q}\right) - v' \cdot q \cdot \frac{t}{q} \right\rfloor$$

$$= \left\lceil \left(\sum_{i=1}^{k} x_i \cdot \left(\tilde{q}_i \cdot \frac{t}{q_i}\right)\right) \right\rfloor - v' \cdot t \;=\; \left[\left\lceil \left(\sum_{i=1}^{k} x_i \cdot \left(\tilde{q}_i \cdot \frac{t}{q_i}\right)\right) \right\rfloor \right]_t$$

- Separate into integer and fractional parts

$$t\tilde{q}_i / q_i \;=\; \omega_i + \theta_i, \quad \text{with } \omega_i \in \mathbb{Z}_t \text{ and } \theta_i \in [-\tfrac{1}{2}, \tfrac{1}{2})$$

- Fractional parts are precomputed and stored as floating-point numbers
- The cost of handling approximation errors to support CRT moduli up to 60 bits is small

RSA Conference2019

Our Results and Their Impact

Experimental Results in PALISADE

Table 1: Timing results for decryption, homomorphic multiplication, and relinearization in the single-threaded mode; $t = 2$, $\log_2 q_i \approx 55$, $\lambda \geq 128$

L	n	$\log_2 q$	k	Dec. [ms]	Mul. [ms]	Relin. [ms]	Multiplication [%]		
							CRT ext.	Scaling	NTT
1	2^{11}	55	1	0.15	3.16	0.41	34	8	52
5	2^{12}	110	2	0.49	10.1	2.58	29	9	56
10	2^{13}	220	4	1.89	38.9	18.7	27	10	56
20	2^{14}	440	8	8.3	174	78.3	27	14	54
30	2^{15}	605	11	25.8	555	332	27	15	52
50	2^{16}	1,045	19	95.8	2,368	2,066	30	20	46
100	2^{17}	2,090	38	409	12,890	16,994	30	20	46

10X FASTER THAN PRIOR BFV IMPLEMENTATION IN PALISADE!

Experimental Results in PALISADE

Table 4: Timing results with multiple threads for decryption, multiplication, and relinearization, for the case of $L = 20, n = 2^{14}, k = 8$ from Table 3

# of threads	Dec. [ms]	Mul. [ms]	Relin. [ms]	Mul. + Relin. [ms]
1	9.83	178.6	95.8	274.4
2	5.90	114.1	53.8	168.0
3	4.93	79.5	49.6	129.1
4	3.92	66.3	37.4	103.7
5	3.95	58.7	38.8	97.5
6	4.07	52.2	40.2	92.4
7	4.01	49.9	38.9	88.8
8	3.13	43.3	29.2	72.5
9	3.17	38.0	31.4	69.5
16	3.37	34.9	32.7	67.6
17	3.46	32.0	33.2	65.2
32	3.47	29.2	33.1	62.4

Other Applications of Our Work

- The RNS operations proposed in our work can also be used for CKKS and BGV, as well as many other number theory cryptographic primitives.
- For instance, they were used to develop an efficient RNS variant of CKKS for a winning secure genome-wide association studies (GWAS) solution at iDASH’18.
  - For 245 individuals, 15K SNPs (genetic variations), and 3 covariates Duality Technologies developed a logistic-regression-based HE solution in PALISADE that runs under 4 minutes on a 4-core machine and uses less than 10 GB of RAM.

Apply Our BFV Variant to Your Problem!

- Download PALISADE library
  - palisade-crypto.org
- Download the manual
  - https://git.njit.edu/palisade/PALISADE/blob/master/doc/palisade_manual.pdf
- Write an HE-enabled version of your application
- Contact us by email if you have any questions
  - palisade@njit.edu

References

[GHS12] Gentry C., Halevi S., Smart N.P. (2012) Homomorphic Evaluation of the AES Circuit. CRYPTO 2012.

[CR14] D. B. Cousins and K. Rohloff, A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU, WAHC'14.

[DHS16] Yarkın Doröz, Yin Hu, and Berk Sunar, Homomorphic AES evaluation using the modified LTV scheme, Designs, Codes and Cryptography, Vol. 80, 2016.

[BEHZ16] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca, A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes, SAC'16.