Evaluation of Homomorphic Primitives for Computations on Encrypted Data for CPS systems

Peizhao Hu, Tamalika Mukherjee, Alagu Valliappan and Stanisław Radziszowski
Department of Computer Science
Rochester Institute of Technology, USA
Email: {hxpvcs, txm1809, aav5705, sprics}@rit.edu

***Abstract*—In the increasingly connected world, cyber-physical systems (CPS) have been quickly adapted in many smart city applications, such as smart grids or healthcare. There will be more and more highly sensitive data important to the users being collected and processed in the cloud computing environments. Homomorphic Encryption (HE) offers a potential solution to safeguard privacy through cryptographic means while allowing the service providers to perform computations on the encrypted data. Throughout the process, only authorized users have access to the unencrypted data. In this paper, we provide an overview of three recent HE schemes, analyze the new optimization techniques, conduct performance evaluation, and share lessons learnt from the process of implementing these schemes. Our experiments indicate that the YASHE scheme outperforms the other two schemes we studied. The findings of this study can help others to identify a suitable HE scheme for developing solutions to safeguard private data generated or consumed by CPS.**

I. Introduction

The rise of ubiquitous connectivity, Internet of Things (IoT), cloud computing and big data analytics have boosted the rapid growth of CPS. In this increasingly connected world, everyday IoT objects collect and analyze vast amounts of information about us in order to tailor their services to better suit our needs and intentions. While the advantage of this new computing era is prominent, it poses serious privacy concerns due to the sensitivity of the data being collected, and how and where this data is processed [1], [2]. In typical CPS systems, private data is transmitted and processed in a cloud computing environment. This means our private data is vulnerable to various attacks and security breaches [3], which are quite common.

Classical cryptography techniques, such as public-key cryptosystems or the AES (Advanced Encryption Standard), are commonly used for protecting this sensitive data in network transmission and cloud storage. Data encrypted under these classical methods cannot be processed further without decryption. This limits the potential benefit of aggregating crowd-sourced data to derive critical information, for example, dynamic power distribution based on demands [4]. Recent advances in Homomorphic Encryption (HE) [5] may provide a solution to this problem. HE is a cryptographic technique that preserves privacy through encryption while supporting computations over encrypted data. In a nutshell, for messages m and m' we want the following properties to hold for encryption using a key k:

$$Enc(m,k) + Enc(m',k) = Enc(m+m',k),$$
$$Enc(m,k) * Enc(m',k) = Enc(m*m',k),$$

where applying addition $+$ and multiplication $*$ to ciphertexts has the same effect as applying these operations to plaintexts and then encrypting the results. Since all functions can be broken down into these basic operations, we could theoretically construct Fully Homomorphic Encryption (FHE) schemes that perform arbitrary computations on encrypted data. The state of the art FHE schemes are mostly constrained by performance issues and large ciphertext size [6]. Hence, we focus our discussion on SomeWhat Homomorphic Encryption (SWHE) schemes, which support computations on encrypted data up to certain number of multiplications. In recent years, a number of SWHE schemes were proposed [7]. In this paper, we share our experiences and lessons learnt from prototyping three recent HE schemes: NLV [8], FV [9] and YASHE [10]. They are based on the same theoretical construction and share important features such as *key switching* [11], [12]. We develop an evaluation framework that allows us to conduct experiments using the same set of parameters as described in [8]. The evaluation focuses on the performance of homomorphic primitives provided by these three schemes.

Our work is motivated by the lack of appropriate approaches to perform secure computations in the cloud computing environments. Using those homomorphic primitives, such as addition and multiplication, one can develop applications in other domains, including secure information aggregation for smart grids [4], or secure friend finding in social networks [13].

The remainder of this paper is organized as follows. Section II gives a brief overview of homomorphic encryption schemes. We present a performance evaluation of the proposed solutions in Section III. Section IV provides an overview of related work on HE and its applications. Section V concludes the paper and discusses possible future work.

II. Overview of Homomorphic Encryption Schemes

In this section, we discuss three recent SWHE schemes. They base security on the same mathematical hardness and share common feature to control the noise growth after every homomorphic operation. From the original theoretical construction BV [11], [12] to the first implementation NLV [8] based on BV, and to FV [9] and YASHE [10], we will provide an overview of these schemes.

A. Notation

The most important structure is the ring R. Given a positive integer d, we define $R = \mathbb{Z}[x]/(\Phi_d(x))$ as the ring of polynomials with integer coefficients modulo the d-th cyclotomic polynomial $\Phi_d(x) \in \mathbb{Z}[x]$. $\Phi_d(x) = x^n + 1$ when d is a power of 2. For clarity, we fix on this cyclotomic polynomial and denote as $\Phi(x)$ throughout our discussions. An element in this ring is a polynomial of the form $v_0 + v_1 x + \ldots + v_{n-1}x^{n-1}$ where each coefficient v_i is an integer. In order to distinguish between plaintexts and ciphertexts, we use R as the underlying ring structure to define two finite rings: the plaintext space is defined as $R_t = \mathbb{Z}_t[x]/(\Phi(x))$, where $\mathbb{Z}_t$ are integers modulo t, and the ciphertext space is $R_q = \mathbb{Z}_q[x]/(\Phi(x))$, where q is a prime and t is much smaller than q. The messages are encoded as coefficients of polynomials that live in R_t which are of the form $v_0 + v_1 x + \ldots + v_{n-1}x^{n-1}$ where $v_i \in \mathbb{Z}_t$. For example, if $t = 2$, then the message is in binary format, say if $m = 1011$ then the polynomial form of m is $1 + x^2 + x^3$. We can similarly describe the ciphertexts to be elements of the form $v_0 + v_1 x + \ldots + v_{n-1}x^{n-1}$, where $v_i \in \mathbb{Z}_q$ for $0 \le i < n$. We also define a Gaussian distribution χ_e on R which we use to introduce noise (error term) into the ciphertexts. For the error term, which is in the form of a polynomial, we sample its coefficients from χ_e independently for each coefficient. Note that the error term should be small, hence we use a Gaussian distribution that is centered at zero and has a small standard deviation. Finally, we represent modulo operations, such as (v mod t) as $[v]_t$.

B. Ring-Learning with Error (Ring-LWE)

Cryptographic schemes typically base their security on some kind of mathematical hardness problems, like approximate greatest common divisor (GCD) [14], [15] and LWE [16]. The SWHE schemes that we are dealing with are based on the assumption that the Ring-LWE problem is as hard as certain lattice problems [16], [17]. The Ring-LWE assumption is stated as follows:

If we uniformly sample s and a_i from a ring $R_q = \mathbb{Z}_q[x]/(\Phi(x))$ and e_i from a Gaussian distribution χ_e, such that $b_i = a_i s + e_i$ for $i \in \mathbb{N}$, then b_i is *computationally* indistinguishable from elements that are uniformly sampled from R_q. In layman terms we hide secret element s covering it with normal distribution of elements in R_q.

C. Recent SWHE schemes

In this paper, we review three recent SWHE schemes: NLV [8], FV [9] and YASHE [10]. We prototype these schemes to construct HE primitives for performing homomorphic proximity computations. For each of these schemes we explain the three main cryptographic components: *key generation*, *encryption* and *decryption*. In addition, we describe how homomorphic operations are achieved in these schemes. Note that we focus on the asymmetric scheme, hence we need a secret key SK and the corresponding public key PK.

1) NLV scheme: The NLV scheme [8], proposed by Naehrig, Lauter and Vaikuntanathan, is one of the constructions based on the BV SWHE scheme [11], [12]. It is a straightforward construction as many operations are based on the Ring-LWE assumption we have discussed.

a) Key Generation: For a secret key $SK = s$, we sample its coefficients from a Gaussian distribution χ_k, denoted by $s \leftarrow \chi_k$, a random element $a_1 \in R_q$ and an error $e \leftarrow \chi_e$. This way we only need to keep a relatively small private key. To improve security, χ_k is different from χ_e in mean and/or standard deviation. We set the public key to be $PK = (a_0, a_1)$, where $a_0 = -(a_1 \cdot s + t \cdot e)$ and t is the modulus of the plaintext space. Note that s, a_0, a_1 and e are all elements of ring R_q.

b) Encryption: Given a plaintext $m \in R_t = \mathbb{Z}_t[x]/(\Phi(x))$ and a public key $PK = (a_0, a_1)$, we construct an encryption function $Enc(m, PK) = (c_0, c_1) = (a_0 e_1 + te_2 + m, a_1 e_1 + te_3) \in (R_q)^2$, where $e_i, i = 1, 2, 3$ are noises sampled independently from the Gaussian distribution χ_e.

c) Decryption: While any fresh encryption will produce a ciphertext with two components $C = (c_0, c_1) \in (R_q)^2$, homomorphic multiplication (described below) will increase the number of elements in the ciphertext beyond two. Hence, we represent the ciphertext as $C = (c_0, \ldots, c_\xi) \in (R_q)^{\xi+1}$. The decryption function is defined as $Dec(C, SK) = \tilde{m} = \sum_{i=0}^{\xi} c_i s^i \in R_q$.

To understand its correctness, we show the following proof for a ciphertext with two elements $C = (c_0, c_1)$:

$$\begin{aligned}\sum_{i=0}^{1} c_i s^i &= (a_0 e_1 + te_2 + m) + (a_1 e_1 + te_3)s \\ &= -a_1 e_1 s - te_1 e + te_2 + m + a_1 e_1 s + te_3 s \\ &= t(-e_1 e + e_2 + e_3 s) + m.\end{aligned}$$

The coefficients of the resulting expression must be converted from $(0, q]$ to $(-q/2, q/2]$ in order to properly represent the error terms, since they are drawn from Gaussian distribution χ_e. We should then be able to decrypt the plaintext m from $[\tilde{m}]_t$, given that the noise terms are small.

d) Homomorphic Operations: Given two ciphertexts $C = (c_0, \ldots, c_\xi)$ and $C' = (c'_0, \ldots, c'_\eta)$, the homomorphic addition is a straightforward component-wise addition.

$$C + C' = (c_0 + c'_0, \ldots, c_\xi + c'_\eta) \in (R_q)^{max(\xi,\eta)+1},$$

where we might need to pad the ciphertexts by 0's in order to match the length of the longer ciphertext.

Homomorphic multiplication is more difficult, because of the growth of elements,

$$C * C' = (\hat{c}_0, \ldots, \hat{c}_{\xi+\eta}).$$

where $\hat{c}_i$ are appropriate convolutions defined in [11], [12]. In a nutshell, homomorphic multiplication introduces terms with s^i, for $i > 1$. Take the case of multiplying two ciphertexts of length two: $C = (c_0, c_1)$ and $C' = (c'_0, c'_1)$. We want $C * C' = mm' + te_{mult}$ so that we get back mm' (mod t) where m and m' are the corresponding messages, and

e_{mult} is the error resulting from multiplying two ciphertexts. Working backwards, since we know that $m = c_0 + c_1 s$ and $m' = c'_0 + c'_1 s$, we have:

$$\begin{aligned} mm' + te_{mult} &= (c_0 + c_1 s)(c'_0 + c'_1 s) \\ &= c_0 c'_0 + (c_0 c'_1 + c_1 c'_0)s + c_1 c'_1 s^2. \end{aligned}$$

Thus we have $C * C' = (\hat{c}_0, \hat{c}_1, \hat{c}_2) = c_0 c'_0 + (c_0 c'_1 + c_1 c'_0)s + c_1 c'_1 s^2$, where $\hat{c}_0 = c_0 c'_0$, $\hat{c}_1 = c_0 c'_1 + c_1 c'_0$ and $\hat{c}_2 = c_1 c'_1$. A new term with s^2 is introduced. There is a technique, called "*relinearization*", to reduce the number of ciphertext terms. To reduce this three-element ciphertext back to a two-element ciphertext, we construct a set of evaluation keys $Eval_i = (b_i = -(a_i s + te_i) + t^i s^2), a_i)$ for $i = 0, \ldots, \lceil log_t q \rceil - 1$ (i.e., when $t = 2$, $\lceil log_t q \rceil$ is the bit length of q, or *Bit-Decomposition*). The evaluation keys introduce the term s^2 for converting the ciphertext back to a two element ciphertext; so that, the ciphertext is decryptable by the original secret key s. Interested readers can find more information in [8].

2) FV scheme: Based on the NLV construction, Fan and Vercauteren [9] proposed the FV scheme with scale invariance $\Delta = \lfloor q/t \rfloor$ to control the noise growth after every homomorphic multiplication. Since we have $q = \Delta \cdot t + [q]_t$, we remark that q and t do not have to be prime, nor that t and q are coprime.

a) Key Generation: The key generation is almost the same as in the NLV scheme. We generate a secret key $SK = s \leftarrow \chi_k$. For a random element $a_1 \in R_q$ and an error $e \leftarrow \chi_e$, we set the public key to be $PK = (a_0, a_1)$, where $a_0 = [(-a_1 \cdot s + e)]_q$.

b) Encryption: The encryption is also similar to the NLV scheme. Given a plaintext $m \in R_t$ and a public key $PK = (a_0, a_1)$, we construct an encryption function $Enc(PK, m) = (c_0, c_1) = (a_0 \cdot e_1 + e_2 + \Delta \cdot m, a_1 \cdot e_1 + e_3) \in (R_q)^2$, where $e_i, i = 1, 2, 3$ are noises sampled independently from the Gaussian distribution χ_e. It should be noted that Δ is only applied to message m.

c) Decryption: The decryption function is defined as $Dec(C, SK) = \tilde{m} = [\lfloor t/q \rceil \cdot \sum_{i=0}^{\xi} [c_i s^i]_q]_t$ for ciphertext $C = (c_0, \ldots, c_\xi) \in (R_q)^{\xi+1}$. Here, we reduce all noise terms by a factor approximate to $\lfloor t/q \rceil$, which is the inverse of Δ. This scaling down will not have effect on message m since we scale it up by Δ in encryption.

d) Homomorphic Operations: The homomorphic addition and multiplication are the same as in the NLV scheme. The same *relinearization* technique is used in FV to make ciphertext decryptable by the original secret key s. The only difference is that each resulting ciphertext component is multiplied by $\lfloor t/q \rceil$ after each homomorphic operation to scale down the noise. Interested readers can find more information in [9].

3) YASHE: In [18], Stehlé and Steinfeld modified *NTRU-Encrypt* scheme to reduce security to standard problem in ideal lattices. López-Alt, Tromer and Vaikuntanathan constructed a Fully HE scheme based on this modified system [19], however a non-standard assumption is required to allow homomorphic operations and prove security. Bos et al. [10] proposed YASHE (Yet Another Somewhat Homomorphic Encryption) scheme in which this non-standard assumption is removed via a tensoring technique introduced by Brakerski [20]. YASHE is a new Fully HE scheme based on standard lattice assumption and a circular security assumption.

a) Key Generation: In YASHE, the key generation is based on NTRU system. The noise elements f' and g are sampled from χ_k, we find $f = [tf' + 1]_q$ such that f is invertible modulo q. If we find a f satisfied by f', we define $SK = f$, and we then define $PK = h = [tgf^{-1}]_q$.

b) Encryption: Given a plaintext $m \in R_t$, sample e_1, e_2 from χ_e. The corresponding ciphertext is given by

$$Enc(PK, m) = C = [\Delta[m]_t + e_1 + he_2]_q$$

.

c) Decryption: Given a ciphertext $C \in R_q$, we decrypt by computing $Dec(C, SK) = \tilde{m} = \left[\lfloor \frac{t}{q} \cdot [fc]_q \rceil \right]_t \in R_t$.

d) Homomorphic Operations: The homomorphic operations are very straightforward in YASHE since the encryption will produce one ciphertext component as a result of the NTRU key generation. However, multiplying two ciphertexts still results in a quadratic expression. A technique, called *key switching* (similar to *relinearization*) is applied after each homomorphic multiplication to make the ciphertext decryptable by the secret key f. The evaluation key in YASHE is generated as $Eval_i = f^{-1} P(D(f) \otimes D(f)) + e_1 + h \cdot e_2$ for error terms $e_i, i = 1, 2$. In this equation, $P(x)$ and $D(x)$ are PowerOfTwo and BitDecomposition for the plaintext space $t = 2$. $\otimes$ is scalar product of the vectors. For more details, refer to the next section or in [10].

D. Discussion

When comparing these three schemes, scale invariance was introduced in FV and YASHE for controlling the noise growth after every homomorphic multiplication. *Relinearization* (or *key switching*) is used by all schemes to convert the quadratic component as a result of the multiplication. The use of NTRU key generation reduces the space requirement on public and private key pair in YASHE. Subsequently, the encryption in YASHE produces one ciphertext element instead of two in NLV and FV. This reduce the ciphertext size to almost half assuming the message is relatively smaller than the ciphertext space R_q. It has been shown that although the noise growth is smaller in FV, YASHE is faster in performance than FV [6]. In the later section, we conduct experiments to evaluate their performance on three platforms.

III. Evaluation and Discussion

In this section, we describe our SWHE framework in which we implemented the three SWHE schemes. This follows by the evaluation of main homomorphic operations.

A. Implementation and evaluation platforms

To conduct performance evaluation of these SWHE schemes, we have developed a HE prototyping framework that provides functionalities such as time measurement, parameter recording and playback. We implemented the three schemes in C++ with the support for polynomial operations from the Number Theory Library (NTL) version 9.4.0, which depends on the GNU Multiple Precision Arithmetic Library (GMP) version 6.1.0 to handle large integers. Main functionalities of the individual schemes were implemented. We verified the correctness of our implementation through extensive validation tests, and we compared our performance results with the data reported in the original papers. Computation time measurements were done on a 2.6 GHz Intel Core i5 computer.

B. Evaluation results and discussion

In HE schemes, parameter selection is an important process that determines the correctness, security and performance of the schemes. We conduct experiments using the parameters described in Table 1 in [8]. The range of parameters selected for the experiments cover different degree of strength against the distinguishing attack [21].

Figure 1 shows the performance of the NLV scheme. We use these results as the baseline for the performance evaluation of two other HE schemes. As mentioned in Section II, While the NLV is the first implementation of the BV scheme, the FV scheme adds scale invariants to reduce noise growth, and YASHE carries on the use of scale invariants and makes use of the NTRU key generation to reduce the number of ciphertext element. As shown in the Figure, the computation times grow when the parameters increase. But multiplication with relinearization and the processes for generating the evaluation key for relinearization substantially increase the computation time. The evaluation key is a set of $log_t q$ sub-keys; each of such sub-keys is a partial encryption of the secret key with different noises. Hence, generating the evaluation key takes substantial time. Each homomorphic multiplication takes approximately 4.5 times longer than encryption. However, the relinearization step add significant longer time to multiplication. We note that time for decryption after a relinearization step does not increase. These results highlight the needs of new method to improve or replace the relinearization step.

Figure 2 shows how FV and YASHE perform when compared with NLV. One obvious observation is FV has similar or worser performance than the NLV scheme. This indicates that the scale invariants may not provide much of the performance gain over NLV. We will conduct further experiments to investigate this issue in the future work. But it is certain that the scale invariants do add significant computation times when performing multiplication with relinearization. In fact, this technique is applied after every multiplication.

Another observation is that YASHE takes significant longer time in generating the secret key comparing to NLV and FV. This is because NLV and FV sample the secret key terms from the Gaussian distribution χ_e, whereas YASHE's NTRU key generation will try to find suitable f' such that f is invertible modulo q, as described in Section II. In general, YASHE outperforms the other two schemes in computation times, except generating the secret key and performing decryption. The use of NTRU keys produces single ciphertext element instead of two in NLV and FV. For this reason, we conjuncture that YASHE will be a better choice if ciphertext size does matter.

t	D	n	log q	SK	PK	Eval_k	Enc	Dec.deg1	Add	Mult	Mult+Relin	Dec.deg2	Overall
				ms	ms	ms	ms	ms	ms	ms	ms	ms	s
2	1	512	19	0.15	1.23	35.12	1.55	1.21	0.05	-	-	-	0.111
	2	1024	38	0.24	3.38	164.21	4.00	2.79	0.11	15.33	379.08	2.74	0.576
	3	2048	64	0.48	7.89	783.35	11.61	7.69	0.24	43.42	1957.93	7.14	2.833
	4	2048	89	0.50	8.64	1256.50	12.93	9.18	0.23	51.34	3466.72	8.15	4.823
	4	4096	94	0.95	17.69	2903.32	28.74	18.31	0.49	104.59	7759.26	17.22	10.840
	5	4096	120	1.02	20.59	4337.59	37.08	19.89	0.50	138.68	13590.35	22.74	17.711
	10	8192	264	2.07	91.92	45951.80	156.12	93.58	1.42	645.06	121468.00	82.18	164.753
	15	16384	423	4.84	294.64	228059.00	542.52	265.35	4.08	1923.99	636690.00	291.21	849.121
128	1	1024	27	0.24	2.25	14.39	3.19	3.09	0.11	-	-	-	0.075
	2	2048	52	0.49	6.96	86.62	10.30	6.44	0.32	40.91	144.92	6.44	0.316
	3	2048	82	0.49	7.77	157.05	15.22	7.44	0.22	45.93	271.95	7.56	0.521
	3	4096	86	1.01	17.85	384.04	32.44	17.62	0.53	112.41	608.45	16.64	1.229
	4	4096	118	1.03	20.44	636.59	41.69	22.38	0.49	135.30	939.78	19.96	1.861
	5	4096	150	1.04	26.15	892.10	39.56	23.04	0.49	150.92	1316.59	21.57	2.522
	10	8192	324	2.10	113.11	8865.39	207.10	106.52	1.57	788.32	12518.00	102.25	22.807
	10	16384	338	4.30	252.37	21317.95	461.26	245.71	4.42	1795.20	29099.55	285.58	53.558
	15	16384	513	5.20	352.57	45387.65	651.98	313.58	6.96	2481.60	60672.40	331.09	110.583
1024	1	1024	30	0.28	2.82	11.65	5.21	3.69	0.10	-	-	-	0.072
	2	2048	58	0.60	6.96	62.29	9.37	7.31	0.22	37.46	115.17	6.05	0.259
	3	2048	91	0.56	8.56	143.86	13.21	8.24	0.22	57.90	222.22	8.66	0.482
	3	4096	95	0.94	19.00	294.79	31.67	17.25	0.45	111.09	510.52	23.07	1.047
	4	4096	130	0.98	22.20	476.71	34.40	23.23	0.72	141.06	726.70	19.31	1.484
	5	4096	165	1.07	27.97	876.07	49.64	26.75	0.54	183.16	1185.75	27.30	2.426
	5	8192	171	2.02	60.39	1825.48	101.31	57.95	1.15	391.72	2596.47	55.19	5.214
	10	8192	354	2.08	113.49	6837.88	185.85	102.33	1.93	785.03	9213.68	101.98	17.541
	10	16384	368	4.79	257.20	16706.90	462.41	241.67	5.08	1772.87	21488.45	241.25	41.520
	15	16384	558	5.46	389.85	39562.15	715.41	380.39	6.42	2988.03	48260.75	346.22	92.859

Fig. 1. Computation times of different homomorphic operations in the NLV scheme. [t] plaintext space; [D] number of multiplications supported plus one, D=1 implies no multiplication can be performed; [n] degree of a polynomial; [log q] size of the ciphertext space; [SK] secret key; [PK] public key; [Eval_k] evaluation key for relinearization; [Enc] encryption; [Dec.deg1] decryption without relinearization; [Dec.deg2] decryption after one relinearization; [Add] addition; [Mult+Relin] multiplication with relinearization; [Overall] overall time. All values of the individual operations are represented in milliseconds, while overall time is in seconds.

IV. Related work

There are many theoretical constructions of HE schemes, among which only some have practical implementations. Performance evaluation of such schemes so far has been limited. In [6], Lepoint and Naehrig implemented two HE schemes, FV and YASHE, using the FLINT library. Their focus was on the performance of these two schemes on evaluating the operations of a lightweight block-cipher SIMON [22]. The authors compare and contrast the noise growth in homomorphic multiplication both theoretically and experimentally. This paper differs from their work twofolds: (i) analysis of new techniques, such as scale invariants and NTRU key generation, that arguably make FV and YASHE better than the NLV scheme, and (ii) performance evaluation of three schemes implemented in NTL library with focus on improvement provided by the new techniques.

V. Conclusion

The growing popularity of IoT and CPS systems has paved the way for computers to "disappear" in the background of our life and provide services that suit our needs in smart environments. Large amount of sensor data is collected and processed in the cloud to derive better understanding about the

t	D	n	log q	SK (ms)		PK (ms)		Eval_k (ms)		Enc (ms)		Dec.deg1 (ms)		Add (ms)		Mult+Relin (ms)		Dec.deg2 (ms)		Overall (s)	
				FV	YASHE	FV	YASHE	FV	YASHE	FV	YASHE	FV	YASHE	FV	YASHE	FV	YASHE	FV	YASHE	FV	YASHE
2	1	512	19	0.01	-7.55	-0.24	0.55	1.12	21.00	-0.17	0.77	-0.31	-0.25	-0.03	0.03	-	-	-	-	-0.041	0.042
	2	1024	38	-0.01	-24.84	0.47	1.24	-15.92	90.11	-1.15	1.50	-1.08	-0.79	-0.05	0.08	-72.62	378.81	-0.64	-0.46	-0.174	0.194
	3	2048	64	-0.05	-72.48	0.03	2.20	-96.57	399.90	0.05	6.02	-1.76	-0.86	-0.11	0.16	-372.04	1956.45	-14.82	-1.19	-0.675	1.355
	4	2048	89	0.01	-81.50	-0.69	0.39	-88.51	663.44	-1.22	6.38	-0.62	-0.08	-0.11	0.15	-534.10	3464.09	-1.61	-1.95	-0.783	1.485
	4	4096	94	-0.12	-185.05	-2.56	2.64	-2.51	1523.37	0.70	14.81	-4.21	-1.97	-0.19	0.33	-1650.81	7752.25	-2.50	-4.15	-2.417	2.201
	5	4096	120	-0.08	-283.84	-1.91	1.89	-185.79	2022.99	4.80	18.00	-4.13	-4.89	-1.17	0.34	-350.25	13580.03	-35.03	-5.62	-1.723	4.697
	10	8192	264	-0.02	-1243.03	-1.60	14.42	3728.75	26058.40	-3.75	76.45	-2.49	-0.12	-1.00	0.99	-7345.50	121378.80	-149.98	-5.27	-8.200	54.078
	15	16384	423	-2.95	-4360.76	-232.14	50.95	10344.00	122910.00	-34.08	304.89	-57.96	-3.63	-10.30	2.87	-36438.00	636229.59	-488.81	-22.54	-54.469	278.133
128	1	1024	27	-0.01	-17.53	-0.13	0.77	-0.91	7.27	-0.01	-	-0.85	-	-0.05	-	-	-	-	-	-0.138	0.011
	2	2048	52	-0.02	-70.41	-2.22	0.24	6.21	48.16	-0.21	5.11	-2.46	-1.29	-0.09	0.16	-251.06	-	-10.51	-	-0.486	0.044
	3	2048	82	0.01	-74.81	0.15	2.54	13.95	54.70	3.59	8.23	-1.73	-1.22	-0.12	0.15	-194.19	271.79	-12.40	-2.85	-0.436	0.143
	3	4096	86	-0.03	-185.10	0.90	4.88	-156.97	205.14	4.25	17.27	-18.94	-1.49	-0.44	0.36	-1273.44	607.96	-63.57	-3.79	-2.425	0.299
	4	4096	118	-0.36	-235.36	-0.64	4.89	27.71	350.72	6.00	25.02	-2.84	-0.77	-0.38	0.19	-995.00	939.10	-36.75	-6.08	-1.950	0.584
	5	4096	150	-0.12	-304.85	0.46	7.70	-79.66	455.45	-4.69	17.96	-14.79	-2.07	-0.77	0.32	-1013.45	1315.76	-43.23	-3.18	-2.148	0.860
	10	8192	324	0.06	-1601.23	-8.53	21.92	316.03	4360.60	20.61	114.40	-1.82	-3.42	-1.51	1.13	-3651.45	12512.43	-177.21	-8.99	-7.170	10.722
	10	16384	338	-1.16	-3763.31	-5.21	42.21	-265.90	10735.75	-30.59	248.37	-32.28	10.49	-1.53	3.40	-20508.80	29084.62	-384.05	32.20	-40.011	23.369
	15	16384	513	-0.43	-5401.17	-193.54	42.00	-1803.30	22892.25	-28.97	336.70	-29.16	-53.82	-3.35	5.62	-31357.65	60643.35	-577.41	-19.28	-60.183	52.285
1024	1	1024	30	0.04	-19.42	-0.10	0.93	1.21	6.66	0.93	3.61	-0.25	-1.22	-0.05	0.07	-	-	-	-	-0.142	0.002
	2	2048	58	0.09	-60.03	-0.01	2.76	4.71	31.23	0.67	4.89	-0.68	-0.21	-0.13	0.15	-226.28	115.06	-9.68	-1.31	-0.459	0.027
	3	2048	91	0.08	-87.87	-0.10	2.36	8.64	78.21	0.30	6.70	-1.99	-1.34	-0.12	0.09	-228.45	222.04	-12.68	-0.83	-0.456	0.113
	3	4096	95	-0.09	-235.40	1.31	-6.46	4.61	155.36	-2.93	5.88	-3.36	-4.97	-0.46	0.29	-878.19	510.10	-21.99	3.09	-1.831	0.152
	4	4096	130	-0.05	-238.13	-4.26	4.97	30.50	166.32	1.17	-5.67	-0.46	-4.82	-0.04	0.53	-1020.68	726.09	-43.30	19.12	-1.918	0.213
	5	4096	165	0.03	-377.73	-4.09	3.34	83.68	476.98	5.16	26.61	-5.34	-3.39	-0.41	0.35	-1070.33	1184.90	-52.86	-2.64	-1.901	0.687
	5	8192	171	0.07	-766.21	1.54	9.79	-2.98	840.33	0.43	43.21	-10.74	-67.99	-0.77	0.79	-3961.58	2594.46	-100.67	-7.68	-7.752	1.155
	10	8192	354	-0.03	-2637.15	3.65	20.87	-609.77	3161.30	-3.28	85.48	-12.56	-15.87	-1.25	1.34	-4344.53	9205.46	-13.62	-89.12	-8.291	2.504
	10	16384	368	-0.43	-3641.10	-62.17	46.90	-275.30	8731.47	-44.85	250.34	-19.17	1.86	-2.78	4.02	-19773.20	21476.73	-426.31	-26.48	-39.202	17.247
	15	16384	558	-0.74	-9667.39	-11.45	-54.07	1006.35	18268.85	-50.39	289.20	-150.12	10.88	-3.92	4.94	-39445.80	48235.82	-712.75	-47.84	-63.008	35.331

Fig. 2. Difference in computation times using NLV against FV and YASHE. All values shown in this table are calculated using NLV as the baseline. For example, the values for FV columns are calculated as $Value_{NLV} - Value_{FV}$. Hence, larger the values are the better.

environments and users. Concerns around privacy is a major barrier before this computing era to reach its full potential. Homomorphic Encryption has be considered as a potential tool to address this problem. In this paper, we provided an overview of three recent HE schemes and conducted performance evaluation of main homomorphic operations. We shared the lessons learnt from implementing these schemes.

References

[1] Federal Trade Commission, "Internet of things: Privacy and security in a connected world," http://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf, January 2015.

[2] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview, understanding the issues and challenges of a more connected world," The Internet Society, Tech. Rep., October 2015.

[3] R. A. Popa and N. Zeldovich, "How to compute with data you can't see," IEEE Spectrum, July 2015.

[4] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Proceeding of SmartGridComm 2010*, Gaithersburg, MD, USA, Oct 2010, pp. 327–332.

[5] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[6] T. Lepoint and M. Naehrig, "A comparison of the homomorphic encryption schemes fv and yashe," in *IACR Cryptology ePrint Archive, 2014:062*, 2014.

[7] V. Vaikuntanathan, "Computing blindfolded: New developments in fully homomorphic encryption," in *In Proceedings of FOCS '11*, Oct 2011, pp. 5–16.

[8] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of CCSW '11*, 2011, pp. 113–124.

[9] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," https://eprint.iacr.org/2012/144/20120322:031216, March 2012.

[10] J. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Cryptography and Coding*, ser. LNCS, M. Stam, Ed. Springer Berlin Heidelberg, 2013, vol. 8308, pp. 45–64.

[11] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) lwe," in *Proceedings of FOCS '11*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 97–106.

[12] ——, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Proceedings of CRYPTO'11*. Santa Barbara, CA: Springer-Verlag, 2011, pp. 505–524.

[13] B. Niu, Y. He, F. Li, and H. Li, "Achieving secure friend discovery in social strength-aware pmsns," in *Military Communications Conference, MILCOM 2015 - 2015 IEEE*, Oct 2015, pp. 947–953.

[14] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Proceedings of CRYPTO'11*. Santa Barbara, CA, USA: Springer-Verlag, 2011, pp. 487–504.

[15] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proceedings of EUROCRYPT'10*. French Riviera, France: Springer-Verlag, 2010, pp. 24–43.

[16] O. Regev, "The learning with errors problem (invited survey)," in *Proceeding of CCC'10*, Cambridge, MA, June 2010, pp. 191–204.

[17] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *J. ACM*, vol. 60, no. 6, pp. 43:1–43:35, Nov. 2013.

[18] D. Stehlé and R. Steinfeld, "Making ntru as secure as worst-case problems over ideal lattices," in *Advances in Cryptology – EUROCRYPT 2011*, ser. Lecture Notes in Computer Science, K. Paterson, Ed. Springer Berlin Heidelberg, 2011, vol. 6632, pp. 27–47.

[19] A. Lopez-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," Cryptology ePrint Archive, Report 2013/094, 2013, http://eprint.iacr.org/.

[20] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," Cryptology ePrint Archive, Report 2012:078, 2012.

[21] R. Lindner and C. Peikert, "Better key sizes (and attacks) for lwe-based encryption," in *Proceedings of CT-RSA 2011*. San Francisco, CA, USA: Springer-Verlag, 2011, pp. 319–339.

[22] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK families of lightweight block ciphers," Cryptology ePrint Archive, Report 2013/404, 2013, http://eprint.iacr.org/.