A Symmetric FHE Scheme Based on Linear Algebra

Iti Sharma
University College of Engineering, Computer Science Department.
itisharma.uce@gmail.com

Abstract— FHE is considered to be Holy Grail of cloud computing. Many applications of cloud computing inherently need symmetric keys while very few FHE schemes have been proposed with symmetric keys. Even with the schemes based on public key cryptography a major hurdle has been the accumulation of noise in ciphertexts leading to limited size of circuits that can be evaluated using the scheme homomorphically. Noise is eliminated through refresh procedure, which has been made essentially possible with public keys containing private keys as a part. Moreover, refreshing of ciphertext is time consuming. We propose here a method of refreshing the ciphertext while not disclosing the secret key, that is, in a symmetric cryptosystem. Also, our scheme doesn't need any evaluation key thus enabling secure delegation of data processing using such scheme.

Keywords— symmetric FHE, fully homomorphic encryption, cloud computing

I. INTRODUCTION

Emerging applications of cloud computing and mobile devices involve heavy computations which instead of running on the lightweight device, are delegated to the cloud, working as a service provider. The goal in this setting is to utilize the resources of a single powerful service provider for the work that computationally weak clients need to perform on their data. This concept of computation-as-a-service accounts for popularity of cloud computing, but has many challenges yet to overcome. A major issue concerns the privacy of user data or user function or both. If the clients' input data is sensitive, we need mechanisms that allow the server to process the data while maintaining its privacy. Cryptographic primitives with homomorphic properties have been proposed to ensure this, but practical Fully Homomorphic Encryption schemes are very few. If the function to be performed over data is a private function (belongs to certain user or cloud), we need mechanisms to prevent other parties from learning anything about the function from the results only. Moreover, it needs symmetric Fully Homomorphic Encryption schemes. This area has gained recent attention and yet an open problem. Homomorphic encryption has largely been studied in context of public key cryptosystems. But there are applications which inherently would require symmetric keys. A symmetric FHE and its applications like private data processing have been proposed in [1] using matrix operations.

Applications like private data processing and secure delegation of computation require capabilities of FHE cryptosystems, but a major concern could arise from the fact that while delegating computation to an untrusted party data can be secured but an evaluation key must be shared so that worker can perform computations homomorphically. Generation and distribution of the evaluation keys is a problem. It is more severe in case of public cryptosystems which involve generation and distribution of public keys also.

Our Contribution: Popular FHE primitives are either based on Gentry's blueprint [2] like [3, 4 and 5] or on learning with errors problem like [6 and 7]. The basic construction follows a method of refreshing long ciphertexts after certain amount of homomorphic computations so that the ciphertext can be decrypted properly. This construction accounts for most of the execution time and is also responsible for large key length. We propose a method to develop symmetric key encryption scheme with fully homomorphic evaluation capabilities based on linear algebra. We propose in this paper a construction for refreshing ciphertext, which is easy and efficient. It derives its security from hardness of factorizing a large integer, which is basis of many public key cryptosystems. Also, the scheme doesn't require an evaluation key.

II. RELATED WORK

Notion of privacy homomorphisms was introduced by Rivest, Adleman and Dertouzos [8]. But it was only in 2009, when Gentry published the first fully homomorphic scheme [2]. Despite the practical limitation of high computational complexity, it ended the long search for the privacy homomorphisms. Gentry's blueprint starts from a Somewhat Homomorphic Encryption scheme which supports limited number of operations to be performed on ciphertext homomorphically due to accumulation of noise in ciphertext at each step of computation. Gentry suggested a "bootstrappability" condition by virtue of which decryption polynomial is of such a lower degree that allows refreshing of ciphertext. Refreshing of ciphertext is in the form of re-encryption, where in the decryption of ciphertext is performed homomorphically (hence secretly) and it is then encrypted under a new key, thus eliminating any noise. This makes the scheme to be fully homomorphic since it can now support arbitrary number of operations. Gentry and Halevi [3] described the first implementation of Gentry's

scheme, establishing the unsuitability of FHE schemes for practical use due to large size of keys and ciphertexts. Van Dijk, Gentry, Halevi and Vaikuntanathan presented a fully homomorphic scheme over the integers [4]. The scheme is conceptually simple in comparison to the original Gentry's scheme; all operations are performed over the integers instead of ideal lattices. However the public-key was too large for any practical system. Coron et al [5] suggested methods to reduce this size of the public key down to $\tilde{O}(\lambda^7)$ from $\tilde{O}(\lambda^{10})$.

Gentry's scheme performs encryption and decryption on plaintext of 1-bit length. Hence, it is intuitive to think that certain operations could be performed on several bits in parallel to reduce runtime. Smart and Vercauteren [9] found that the scheme can be implemented on SIMD(single instruction, multiple data) architecture. Smart and Vercauteren [9] presented methods for selecting parameters for Gentry and Halevi's implementation [3] which use SIMD operations for the somewhat homomorphic scheme. Ways for constructing a fully homomorphic scheme performing re-encryptions in parallel and making SIMD operations useful in practice were also suggested. The parallel version is 2.4 times faster than the standard FHE scheme and the ciphertext size is reduced by a factor of 1/72. Thus, exploiting parallelism in the constituent algorithms can increase efficiency of a scheme.

Major bottleneck in practical deployment of FHE schemes based on Gentry's blue print are large size of keys and high per-gate evaluation time. Brakerski and Vaikuntanathan [6] showed that (leveled) FHE can be based on the hardness of the much more standard "learning with error" (LWE) problem. Brakerski, Gentry and Vaikuntanathan [7] refined the main technique in [6] to construct an FHE scheme with asymptotically linear efficiency. Gupta and Sharma [1] presented an FHE scheme for symmetric encryption based on hardness of factorization of large integer, with operations involving matrix computations. The key size and computation time were reduced enough for practical deployment.

III. GENTRY'S BLUEPRINT AND DGHV SCHEME

Gentry proposed not only the first fully homomorphic encryption scheme, but also a general method to construct fully homomorphic systems [3]. Gentry's construction proceeds in successive steps: first a somewhat homomorphic scheme is used. Secondly Gentry shows how to "squash" the decryption procedure so that it can be expressed as a low degree polynomial in the bits of the ciphertext and the secret key (equivalently a circuit of small depth). Then the breakthrough idea consists in evaluating this decryption polynomial not on the bits of the ciphertext and the secret key (as in regular decryption), but homomorphically on the encryption of those bits. Then instead of recovering the bit plaintext, one gets an encryption of this bit plaintext, i.e. yet another ciphertext for the same plaintext. Now if the degree of the decryption polynomial is small enough, the resulting noise in this new ciphertext can be smaller than in the original ciphertext; this is called the "ciphertext refresh" procedure. Given two refreshed ciphertexts one can apply again the homomorphic operation (either addition or multiplication), which was not necessarily possible on the original ciphertexts because of the noise threshold. Using this "ciphertext refresh" procedure the number of permissible homomorphic operations becomes unlimited and we get a fully homomorphic encryption scheme. The prerequisite for the "ciphertext refresh" procedure is that the degree of the polynomial that can be evaluated on ciphertexts exceeds the degree of the decryption polynomial (times two, since one must allow for a subsequent addition or multiplication of refreshed ciphertexts); this is called the "bootstrappability" condition. Once the scheme becomes bootstrappable it can be converted into a fully homomorphic encryption scheme by providing the encryption of the secret key bits inside the public key.

To gain a deeper understanding, we describe below the scheme due to van Dijk et al [4]. The first step is to describe a Somewhat Homomorphic Encryption scheme. It uses the following four parameters (all polynomial in the security parameter λ):

γ is the bit-length of the integers in the public key,

η is the bit-length of the secret key (which is the hidden approximate-gcd of all the public-key integers),

ρ is the bit-length of the noise (i.e., the distance between the public key elements and the nearest multiples of the secret key), and

τ is the number of integers in the public key.

Also there is a secondary noise parameter $\rho' = \rho + \omega(\log \lambda)$.

KeyGen(λ). The secret key is an odd η-bit integer: $p \xleftarrow{\$} (2\mathbb{Z}+1) \cap [2\eta - 1, 2\eta)$.

For the public key, sample $x_i \xleftarrow{\$} D_{\gamma,\rho}(p)$ for i = 0,…, τ. Relabel so that x_0 is the largest.

Restart unless x_0 is odd and $r_p(x_0)$ is even. The public key is pk = $(x_0, x_1, \ldots, x_\tau)$.

Encrypt(pk, m$\in${0, 1}). Choose a random subset S $\subseteq$ {1, 2, … , τ} and a random integer r in $(-2\rho', 2\rho')$, and output

$$c \leftarrow \left[m + 2r + 2\sum_{i \in S} x_i \right]_{x_0}$$

Evaluate(pk,C, c_1, … , c_t). Given the (binary) circuit *C* with t inputs, and t ciphertexts c_i, apply the (integer) addition and multiplication gates of *C* to the ciphertexts, performing all the operations over the integers, and return the resulting integer.

Decrypt(sk, *c*). Output m′ ← (c mod p) mod 2.

The next step is to "squash" the decryption circuit of the above scheme, namely transform the scheme into one with the same homomorphic capacity but a decryption circuit that is simple enough to allow bootstrapping. For this we need few more parameters κ, θ, Θ, particularly κ = γ, ρ = λ, and Θ= ω(κ·log λ). The modified algorithms are as follows.

KeyGen. Generate sk^* = p and pk^* as before.

Set $x_p \leftarrow \lfloor 2\kappa/p \rceil$, choose at random a Θ -bit vector with Hamming weight θ, $\vec{s} \leftarrow \langle s_1, ..., s_\Theta \rangle$, and let S = {$i : s_i$ = 1}.

Choose at random integers $u_i \in \mathbb{Z} \cap [0, 2^{\kappa+1})$, i = 1,…,Θ, subject to the condition that $\sum_{i \in S} u_i = x_p \pmod{2^{\kappa+1}}$.

Set $y_i = u_i/2^\kappa$ and $\vec{y} \leftarrow \langle y_1, ..., y_\Theta \rangle$ Hence each yi is a positive number smaller than two, with κ bits of precision after the binary point. Also, $\left[\sum_{i \in S} y_i \right]_2 = (1/p) - \Delta_p$ for some $|\Delta_p| < 2^{-\kappa}$.

Output the secret key sk = $\vec{s}$ and public key pk = (pk^*, $\vec{y}$).

Encrypt and Evaluate. Generate a ciphertext c^* as before (i.e., an integer).

Then for i ∈ {1,…,Θ}, set $z_i \leftarrow [c^* \cdot y_i]_2$, keeping only n = ⌈log θ⌉ + 3 bits of precision after the binary point for each z_i. Output both c^* and $\vec{z} \leftarrow \langle z_1, ..., z_\Theta \rangle$.

Decrypt. Output $m' \leftarrow [c^* - \lfloor \sum_i s_i z_i \rceil]_2$.

The scheme is now bootstrappable and fully homomorphic.

IV. PROPOSED SCHEME

We propose a somewhat homomorphic encryption scheme based on the idea from [4] and then proceed to make it fully homomorphic through a refresh procedure which requires an extra key called the refresh key. Encryption and decryption uses symmetric keys while no key is required for evaluation of homomorphic functions over the ciphertexts. The context used is symmetric cryptography that is using only one secret key for both encryption and decryption. Another key called refresh key is used and made public for evaluation but does not reveal the secret key.

A. *Primitives*

Let λ be the security parameter in context of an equivalent computational effort of 2^λ cycles. Message space is (0,1).

The scheme consists of following primitives:

Key generation: Generate the secret key p, a prime number of length λ bits. Select q as refresh key such that it is an even multiple of p, that is q = k p, where k is an even number.

Encryption: Encryption is probabilistic, a many-to-one mapping $\text{Enc} : Z_2 \rightarrow \mathbb{N}$, thus converting the message bit into an integer, and a message may have several encryptions. Encryption involves following steps:

1. Choose m' such that m≡m' mod 2
2. Choose a random number r of length λ^2 bits
3. Output c = m' + pr

Decryption: The decryption needs to be inversion of the encryption process, a mapping $\text{Dec} : \mathbb{N} \rightarrow Z_2$. It is a very simple and efficient operation producing output, the message as m = c mod p mod 2.

Homomorphic Operations: Addition of two plaintext bits, that is XOR operation is homomorphic to direct addition of two ciphertexts (integers). Multiplication of two plaintext bits, that is AND operation is homomorphic to integer multiplication of two ciphertexts.

The above scheme is somewhat homomorphic due to the problem of accumulation of noise. That is after a few operations the noise in ciphertext may exceed beyond the level making its decryption incorrect. Hence we need a refresh procedure to eliminate noise from ciphertext without decrypting. This is beneficial for delegating a computation so that worker can reduce noise in ciphertext in the absence of decryption key. For this purpose we need a refresh key q.

Refresh procedure: Deriving the basic concept of refreshing a ciphertext to make a scheme fully homomorphic from Gentry's blueprint [CG], we design the refresh key so as to contain the secret key in part. But we do not employ the concept of bootstrapping here like in [CG], rather the refresh procedure is very simple and efficient one-step procedure. The refreshed ciphertext is outputted as c' = c mod q.

B. *Comparison with existing schemes*

The factors upon which an FHE scheme can be compared with others are key size, ciphertext size (expansion of plaintext due to encryption) and computation overhead. The overhead is defined as the ratio of the time taken for a computation homomorphically over ciphertext to the time taken to compute on plaintext. We present this comparison of our proposed scheme with the existing popular schemes in Table 1. In context of our scheme, we see the secret key is of length λ bits and refresh key is of length O(λ log λ) bits, which is very smaller as compared to the key sizes in many of the Gentry's blue-print based FHE schemes. The ciphertext size for a one-bit plaintext is dominated by the length of product of p and r. This gives the length of $O(\lambda^2)$ bits. For computation overhead, we take multiplication operation because it is costlier. Consider two plaintexts of size 1 bit. The cost of AND operation is O(1), while homomorphically it is performed as multiplication of two integers of size λ^2 bits, implying effort of $O(\lambda^4)$. The XOR operation performed as addition of two integers is less costly than multiplication operation, hence the computation overhead is dominated by the effort of $O(\lambda^4)$.

TABLE I
COMPARISON OF PROPOSED SCHEME WITH OTHER FHE SCHEMES

Parameters	**FHE Schemes**		
	DGHV	**BGV**	**Proposed Scheme**
Encryption Key Size	$O(\lambda^{10})$	Equal to plaintext	$O(\lambda)$
Computation Overhead	$\tilde{\Omega}(\lambda^{3.5})$	$\tilde{O}(\lambda^2)$	$O(\lambda^4)$
Plaintext Expansion	$O(\log \lambda)$	$O(\lambda^3)$	$O(\lambda^2)$
Encryption Key Size	$O(\lambda^{10})$	Equal to plaintext	$O(\lambda)$

C. *Security of the scheme*

Security is the first and foremost aspect in cryptography. When considering security of certain cryptosystem we aim to find out the amount of effort required by an adversary to discover the key from some amount of other information.

Security of the secret key can be based on the difficulty of factorization of the declared key q, which contains p as factor. Thus, our scheme derives security from hardness of Integer Factorization, similar to many popular cryptosystems.

The brute-force security of our scheme depends on the length of key, hence which is considered sufficiently secure for λ =80.

V. CONCLUSIONS AND FUTURE SCOPE

Cloud computing applications suffer from a major concern about security. This can be ensured through encryption but it doesn't allow for secure delegation of data processing. Here, fully homomorphic encryption is most useful since it allows processing of encrypted data. There is dearth of practically feasible FHE solutions. A few efficient FHE schemes available use public cryptography, while many applications inherently require symmetric keys. Hence, we have proposed a symmetric FHE scheme which works with plaintext bits to convert them into integers using linear algebra. We have proposed a novel method of refreshing ciphertexts to reduce their length after certain number of operations. The security of the proposed scheme is based on the difficulty of large integer factorization.

The scheme can be more generalized to work on integers rather than bits. It would not only lead to an extended scheme rather an improved scheme. Also, parallelization of operations to work on several bits together

can also be a possible extension. Such scheme cannot be directly approached since the primitives involve random numbers which are different for each plaintext. Direct parallelization of operations can only be implemented when performing homomorphic operations. In this case, efficiency improvement will depend on the computation being performed and cannot be considered as a factor of efficiency for the FHE scheme.

REFERENCES

[1] C.Gupta and I. Sharma, A fully homomorphic encryption scheme with symmetric keys with application to private data processing in clouds, Proceedings of Networks of the Future (NOF), 2013 Fourth International Conference. pp.1-4, 23-25 Oct. 2013.

[2] C. Gentry, A Fully Homomorphic Encryption scheme, Dissertation. Sep 2009. Available at https://crypto.stanford.edu/craig/craig-thesis.

[3] C. Gentry and S. Halevi, Implementing Gentry's fully homomorphic encryption scheme, EURO-CRYPT 2011, LNCS, Springer, K. Paterson (Ed.),2011.

[4] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, Fully homomorphic encryption over the integers, Proceedings of Eurocrypt-10, Lecture Notes in Computer Science, vol 6110,. Springer, pp 24-43, 2010.

[5] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. "Fully homomorphic encryption over the integers with shorter public-keys", Advances in Cryptology - Proc. CRYPTO 2011, vol. 6841 of Lecture Notes in Computer Science. Springer, 2011.

[6] Z. Brakerski and V. Vaikuntanathan, Efficient fully homomorphic encryption from (standard) LWE, Foundations of Computer Science, 2011. Also available at Cryptology ePrint Archive, Report 2011/344.

[7] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, Fully homomorphic encryption without bootstrapping, Cryptology ePrint Archive, Report 2011/277.

[8] R. Rivest, L. Adleman, and M. Dertouzos, On data banks and privacy homomorphisms, Foundations of Secure Computation, pp 169-180, 1978.

[9] N. P. Smart and F. Vercauteren, Fully homomorphic SIMD operations, Cryptology ePrint Archive, Report 2011/133.