International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume III, Issue V, May- 2016

A PUBLIC KEY CRYPTO SYSTEM AND SOME WHAT ENCRYPTION USING PALLIER ALGORITHM

S.Suriyaa
Department of Computer Science and Engineering
SSM College of Engineering, Komarapalayam,
Tamil Nadu, India
Suriyaashanmugam@gmail.com

V.Ramya
Department of Computer Science and Engineering
SSM College of Engineering, Komarapalayam
Tamil Nadu, India
ramyaajaagan@gmail.com

Abstract-We proposed a hybrid scheme that combines public key encryption and somewhat homomorphic encryption. The proposed scheme is suitable for cloud computing environments since it has small bandwidth, low storage requirement, and supports efficient computing on encrypted data. Our solution provides a trade-off between the size of the transmitted ciphertexts and the conversion costs. While the ciphertext expansion of PKE is larger than that of AES, it can be homomorphically evaluated with a SHE of much smaller multiplicative depth. The parameters of our hybrid scheme are very large when the message space of the underlying FHE is Z*N*. For an efficient implementation, we need a method to evaluate mod *N* arithmetic using an FHE whose message space is Z*M* for small $M > 2$. The proposed scheme is suitable for cloud computing environments since it has small bandwidth, low storage requirement, and supports efficient computing on encrypted data. We enhance the technique for quickly decrypt the SHE encryption data. To be used in practical applications, homomorphic decryption of the base PKE is too expensive. We accelerate the homomorphic evaluation of the decryption by introducing a method to reduce the degree of exponentiation circuit at the cost of additional public keys. Using same technique, we give an efficient solution to the open problem.

.

I-INTRODUCTION

Cloud computing is the practice of using a network of remote servers that are used to store, manage and process data, rather than a local server or a personal computer. Now a days, cloud computing is one of the dominant methods used for providing computing infrastructure for Internet services. Actually, most cloud computing concepts originally came from the academic research community, but as clouds grew in popularity, industry has driven their design and development. Cloud computing offers computing and software services on demand by connecting to computing resources and access to IT managed services with an ease. This flexibility of cloud based services comes with risk of security and privacy of user's data. Client privacy is a tentative issue as all clients do not have the same demands regarding privacy. In this project, the proposed system provides the cloud data security using the homomorphic encryption technique. This technique provides functionality to perform operations on encrypted data without using the private key

and without decrypting that data. After decrypting the result of this operation, it is the same as if we carried out the calculation on the plain data. This major strength of homomorphic encryption allows the cloud service providers to perform operations on encrypted client data without compromising on the client data privacy. In most real world applications, securing cloud data with efficient homomorphic encryption systems is a work in progress. However, there are a number of practical applications where we use these systems and there is scope for designing more efficient homomorphic encryption systems in the future.

Homomorphic encryption is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This is sometimes a desirable feature in modern communication system architectures. Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services. For example, a chain of different services from different companies could calculate 1) the tax 2) the currency exchange rate 3) shipping, on a transaction without exposing the unencrypted data to each of those services.[1] Homomorphic encryption schemes are malleable by design. This enables their use in cloud computing environment for ensuring the confidentiality of processed data. In addition the homomorphic property of various cryptosystems can be used to create many other secure systems, for example secure voting systems,[2] collision-resistant hash functions, private information retrieval schemes, and many more.

There are several partially homomorphic crypto-systems, and also a number of fully homomorphic crypto-systems. Although a crypto-system which is unintentionally malleable can be subject to attacks on this basis, if treated carefully homomorphism can also be used to perform computations securely.

However, most FHE schemes still have very large ciphertexts (millions of bits for a single ciphertext). This presents a considerable bottleneck in practical deployments. We consider the following situation: several users upload data encrypted with a public-key FHE, a server carries outcomputations on the encrypted data and then sends them to an agency who has a decryption key for the FHE.

This is common in typical FHE scenarios, such as medical and financial applications. In this situation, one approach to reduce the storage requirement is to use AES encryption to encrypt data, and then perform homomorphic computations on ciphertexts after converting to FHE-ciphertexts. This method has a great advantage in storage and communication, because only small AES-ciphertexts are transmitted from user to server, and these are homomorphically decrypted only when their homomorphic computations are required. In an asymmetric setting, we can still use this approach by adding several public-key FHE ciphertexts of a session key.

However this approach is not practical when the amount of messages transmitted simultaneously is small compared with the size of on FHE ciphertext. Moreover, the conversion of AES-ciphertexts into FHE-ciphertexts requires a leveled FHE with multiplicative depth of at least forty. In this paper, we explore an alternative method that

encrypts messages with a public key encryption (PKE) and converts them into SHE-ciphertexts for homomorphic computations. In this approach, the ciphertext expansion ratio is only two or three regardless of the message size. Moreover, the decryption circuit is very shallow when the SHE allows large integers as messages. For example, the decryption circuit of ElGamal over Z*N* has a multiplicative depth of nine under a SHE with the message space Z*N*. We can reduce the depth further by representing the secret exponent *e* as log*w e* binary vectors of length *w*, which is an improvement over the Gentry-Halevi technique.

II-RELATED WORKS

SCALE-INVARIANT FULLY HOMOMORPHIC ENCRYPTION OVER THE INTEGERS

A scale-invariant fully homomorphic encryption scheme based on the LWE problem, in which the same modulus is used throughout the evaluation process, instead of a ladder of moduli when doing \modulus switching". In this paper we describe a variant of the van Dijk et al. FHE scheme over the integers with the same scale-invariant property. Our scheme has a single secret modulus whose size is linear in the multiplicative depth of the circuit to be homomorphically evaluated, instead of exponential; we therefore construct a leveled fully homomorphic encryption scheme. This scheme can be transformed into a pure fully homomorphic encryption scheme using bootstrapping, and its security is still based on the Approximate-GCD problem. We also describe an implementation of the homomorphic evaluation of the full AES encryption circuit, and obtain signi_cantly improved performance compared to previous implementations: about 23 seconds per AES block at the 72-bit (resp. 80-bit) security level on a mid-range workstation. Finally, we prove the equivalence between the (error-free) decisional Approximate-GCD problem introduced by Cheon et al. (Eurocrypt 2013) and the classical computational Approximate-GCD problem. This equivalence allows to get rid of the additional noise in all the integer-based FHE schemes described so far, and therefore to simplify their security proof. Constructed the _rst fully homomorphic encryption scheme (FHE), i.e. a scheme allowing a worker to evaluate any circuit on plaintext values while manipulating only ciphertexts.

SOMEWHAT PRACTICAL FULLY HOMOMORPHIC ENCRYPTION

Fully homomorphic encryption (FHE)

Fully homomorphic encryption (FHE) allows evaluation of arbitrary functions on encrypted data, and as such has a myriad of potential applications such as private cloud computing. Gentry were the _rst to show that FHE is theoretically possible. His construction consisted of three parts: _rst, construct an encryption scheme that is somewhat homomorphic, i.e. that can evaluate functions of limited complexity (think low degree), secondly, simplify the decryption function of this scheme as much as possible (so called squashing), thirdly, evaluate this simpli_ed decryption function homomorphically to obtain ciphertexts with a _xed inherent noise size (so called bootstrapping). The _rst variants of Gentry's scheme all followed the same struc-ture and as such had to make additional security assumptions to enable the squashing

step. More recent schemes avoid squashing all together and can bootstrap by evaluating the real decryption circuit. Another advantage of the more recent schemes is that their security is based on the Learning with Errors (LWE) problem or its ring variant RLWE, the hardness of which can be related to classical problems on (ideal) lattices.

A QUASI-POLYNOMIAL ALGORITHM FOR DISCRETE LOGARITHM IN FINITE FIELDS OF SMALL CHARACTERISTIC

Discrete logarithm problem (DLP)

The discrete logarithm problem (DLP) was first proposed as a hard problem in cryptography in the seminal article of Diffie and Hellman. Since then, together with factorization, it has become one of the two major pillars of public key cryptography. As a consequence, the problem of computing discrete logarithms has attracted a lot of attention. Recently, practical and theoretical advances have been made with an emphasis on small to medium characteristic finite fields and composite degree extensions. The most general and efficient algorithm a complexity of L(1=4 + o(1)) when the characteristic is smaller than the square root of the extension degree. Among the ingredients of this approach, we find the use of a very particular representation of the finite field; the use of the so-called systematic equation1; and the use of algebraic resolution of bilinear polynomial systems in the individual logarithm phase. In this work, we present a new discrete logarithm algorithm, in the same vein as in that uses an asymptotically more efficient descent approach. The main result gives a quasi-polynomial heuristic complexity for the DLP in finite fields of small characteristic. By quasi-polynomial, we mean a complexity of type n O(log n) where n is the bit-size of the cardinality of the finite field. Such a complexity is smaller than any L(_) for _ > 0. It remains super-polynomial in the size of the input, but offers a major asymptotic improvement compared to L(1=4 + o(1)).

ON DATA BANKS AND PRIVACY HOMOMORPHISMS

PRIVACY HOMOMORPHISMS

Encryption is a well—known technique for preserving the privacy of sensitive information. One of the basic, apparently inherent, limitations of this technique is that an information system working with encrypted data can at most store or retrieve the data for the user; any more complicated operations seem to require that the data be decrypted before being operated on. This limitation follows from the choice of encryption functions used, however, and although there are some truly inherent limitations on what can be accomplished, we shall see that it appears likely that there exist encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations. These special encryption functions we call “privacy homomorphisms”; they form an interesting subset of arbitrary encryption schemes (called “privacy transformations”). As a sample application, consider a small loan company which uses a commercial time—sharing service to store its records. The loan company’s “data bank” obviously contains sensitive information which should be kept private. On the other hand, suppose that the information protection techniques employed by the time sharing service are not considered

adequate by the loan company. In particular, the systems programmers would presumably have access to the sensitive information. The loan company therefore decides to encrypt all of its data kept in the data bank and to maintain a policy of only decrypting data at the home office —— data will never be decrypted by the time—shared computer. The situation is thus that of Figure 1, where the wavy line encircles the physically secure premises of the loan company.

FULLY HOMOMORPHIC ENCRYPTION USING IDEAL LATTICES

We propose a fully homomorphic encryption scheme – i.e., a scheme that allows one to evaluate circuits over encrypted data without being able to decrypt. Our solution comes in three steps. First, we provide a general result – that, to construct an encryption scheme that permits evaluation of arbitrary circuits, it suffices to construct an encryption scheme that can evaluate (slightly augmented versions of) its own decryption circuit; we call a scheme that can evaluate its (augmented) decryption circuit bootstrappable. Next, we describe a public key encryption scheme using ideal lattices that is almost bootstrappable. Lattice-based cryptosystems typically have decryption algorithms with low circuit complexity, often dominated by an inner product computation that is in NC1. Also, ideal lattices provide both additive and multiplicative homomorphisms (modulo a public-key ideal in a polynomial ring that is represented as a lattice), as needed to evaluate general circuits. Unfortunately, our initial scheme is not quite bootstrap- pable – i.e., the depth that the scheme can correctly evalu- ate can be logarithmic in the lattice dimension, just like the depth of the decryption circuit, but the latter is greater than the former. In the final step, we show how to modify the scheme to reduce the depth of the decryption circuit, and thereby obtain a bootstrappable encryption scheme, with- out reducing the depth that the scheme can evaluate. Ab- stractly, we accomplish this by enabling the encrypter to start the decryption process, leaving less work for the de- crypter, much like the server leaves less work for the de- crypter in a server-aided cryptosystem.

Disadvantages:

- This scheme is not quite bootstrappable – i.e., the depth that the scheme can correctly evaluate can be logarithmic in the lattice dimension, just like the depth of the decryption circuit, but the latter is greater than the former.

III-EXISTING SYSTEM METHODOLOGY

Several users upload data encrypted with a public-key FHE, a server carries out computations on the encrypted data and then sends them to an agency who has a decryption key for the FHE. This is common in typical FHE scenarios, such as medical and financial applications. In this situation, one approach to reduce the storage requirement is to use AES encryption to encrypt data, and then perform homomorphic computations on ciphertexts after converting to FHE-ciphertexts. This method has a great advantage in storage and communication, because only small AES-ciphertexts are transmitted from user to server, and these are homomorphically decrypted only when their homomorphic computations are required. In an asymmetric setting, we can still use this approach by adding several public-key FHE ciphertexts of a session key. However this

approach is not practical when the amount of messages transmitted simultaneously is small compared with the size of on FHE ciphertext. Moreover, the conversion of AES-ciphertexts into FHE-ciphertexts requires a leveled FHE with multiplicative depth of at least forty.

The result can roughly be broken down into three steps: a general "bootstrapping" result, an "initial construction"using ideal lattices, and a technique to "squash the decryption circuit" to permit bootstrapping. Our research began with the second step: a PKE scheme E1 described in Section 3 that uses ideal lattices and is homomorphic for shallow circuits. A ciphertext has the form v + x where v is in the ideal lattice and x is an "error" or "offset" vector that encodes the plaintext _. Interpreting ciphertext vectors as coefficient vectors of elements in a polynomial ring Z[x]/f(x), we add and multiply ciphertexts using ring operations – 1 + 2 or 1 × 2 – and induce addition and multiplication of the underlying plain-texts. By itself, this scheme improves upon prior work. It compares favorably to Boneh-Goh-Nissim [11]; it is homomorphic for circuits with greater multiplicative depth while allowing essentially unlimited additions. The security of E1is based on a natural decisional version of the closest vector problem for ideal lattices for ideals in a fixed ring.

E1 is homomorphic only for shallow circuits because the "error" vector grows with addition and (especially) multiplication operations; eventually, it becomes so long that it causes a decryption error.

(ElGamal Encryption Over a Ring): Let R be a ring. The *ElGamal encryption scheme* ElG = (ElG.KG, ElG.Enc, ElG.Dec) consists of the following algorithms:

• ElG.KG(λ): Choose a multiplicative cyclic subgroup Gq of prime order q in R such that the DDH assumption
holds with respect to the security parameter λ. Choose a generator g of Gq and a random $e \in [0, q)$, and compute
$y = ge$. Output a public key pkElG = (R, Gq , g, y) and a secret key skElG = e.

• ElG.Enc$(pk$ElG$,m)$: Take as input the public key pkElG and a plaintext $m \in Gq$. Choose a random $r \in [0, q)$ and
compute $g-r$ and $m \cdot yr$. Output $c = (g-r , m \cdot yr)$.

• ElG.Dec$(sk$ElG, $c)$: Take as input the secret key skElG and a ciphertext $c = (v, u) \in R2$. Output $m = veu$.

2.1.1 DISADVANTAGES:

- It not efficient in using cloud environment.
- Take a large space in the storage requirement.
- Very slow process.

IV-PROPOSED SYSTEM METHODOLOGY

In this work explore an alternative method that encrypts messages with a public key encryption (PKE) and converts them into SHE-ciphertexts for homomorphic computations. In this approach, the ciphertext expansion ratio is only two or three regardless of the message size. Moreover, the decryption circuit is very shallow when the SHE allows large integers as messages. For example, the decryption circuit of ElGamal over ZN has a multiplicative depth of nine under a SHE with the message space ZN.2 We can reduce the

depth further by representing the secret exponent *e* as log*w e* binary vectors of length *w*, which is an improvement over the Gentry-Halevi technique. When using additive (resp. multiplicative) homomorphic encryption as the underlying PKEs, we obtain the additional advantage that additions (resp. multiplications) can be computed without converting to SHE.

V-IMPLEMENTATION

A hybrid scheme that combines public key encryption and somewhat homomorphism encryption. It becomes more important to protect the data from misuse by insiders or hacking by outsiders. To reduce the risk, the data may be encrypted prior to storage. Under this scenario, we give an efficient storage solution using a hybrid scheme. Our solution provides a trade-off between the size of the transmitted ciphertexts and the conversion costs. While the ciphertext expansion of PKE is larger than that of AES, it can be homomorphically evaluated with a SHE of much smaller multiplicative depth.

1. **User Login and Data Upload**
2. **User Data convert to Public Key Encryption (PKE)**
3. **PKE data convert to Somewhat Homomorphic Encryption (SHE)**
4. **Upload the file to Storage Environment**

User Login and Data Upload

In this module each user register their information and login to the process. The registered user information can be stored in the app engine backend. After that User uploads their file want to store and file size to the encryption process to reduce the storage requirements.

User Data convert to Public Key Encryption (PKE)

In this module user's data are converted encrypt file by using public key encryption. The public key encryption has been done through ElGamal algorithm. First thing is generation of public key. Then the Process of ElGamal algorithm the key value and user's data are converted into cipher text form. This encrypted file will be used for further process.

PKE data convert to Somewhat Homomorphic Encryption (SHE)

In this module the Public key encryption data convert into somewhat homorphic encrypted data. The SHE (SomeWhat Homomorphic Encryption) key has been generated. This key value and the PHK encrypted file are converted to somewhat homomorphic encryption . Finally we get the SHE ciphertext. This process reduce the bit rate of file size.

Encryption Mechanism

After the upload process completes successfully, user is directed to the encryption page . Here homomorphic encryption is applied on the file containing data using the Paillier algorithm. While the file gets encrypted, there is also functionality to generate a key or a password which can be used while decrypting the same file. As we know that Paillier's algorithm i.e. the additive homomorphic property works only on integer data, we cannot use text other than integers. Also, the product of cipher text which equals to the sum of plain text will not provide any meaning with text

data other than integers. Hence, if we try to execute the Paillier's algorithm on text, we will end up with an output which contains junk data.One way to implement the homomorphism with text would be to convert the text into Big integers and then encrypt. Multiplication can be done with encrypted value of zero so that when we decrypt the final results it will the sum of original Big integer added with zero. This will make sure that we get the original text upon decryption.

Key generation

1. Choose two large prime numbers *p* and *q* randomly and independently of each other such that
   $\gcd(pq,(p-1)(q-1))=1$.
   This property is assured if both primes are of equal length.[1]
2. Compute $n=pq$and
   $\lambda=\operatorname{lcm}(p-1,q-1)$.
3. Select random integer $g$where
   $g\in\mathbb{Z}_{n^2}^*$
4. Ensure $n$divides the order of $g$by checking the existence of the following modular multiplicative inverse:
   $\mu=(L(g^\lambda \bmod n^2))^{-1} \bmod n$,

   where function $L$is defined as
   $L(u)=\frac{u-1}{n}$.
   Note that the notation $\frac{a}{b}$does not denote the modular multiplication of $a$times the modular multiplicative inverse of $b$but rather the quotient of $a$divided by b, i.e., the largest integer value $v\geq 0$to satisfy the relation $a\geq vb$.

- The public (encryption) key is (n,g).
- The private (decryption) key is (λ,μ).

If using p,q of equivalent length, a simpler variant of the above key generation steps would be to set
$g=n+1, \lambda=\varphi(n)$,and
$\mu=\varphi(n)^{-1} \bmod n$, where
$\varphi(n)=(p-1)(q-1)$

Encryption

1. Let $m$be a message to be encrypted where $m\in\mathbb{Z}_n$
2. Select random $r$where $r\in\mathbb{Z}_n^*$
3. Compute ciphertext as:
   $c=g^m\cdot r^n \bmod n^2$

Decryption

Let $c$be the ciphertext to decrypt, where $c\in\mathbb{Z}_{n^2}^*$

1. Compute the plaintext message as:
   $m=L(c^\lambda \bmod n^2)\cdot\mu \bmod n$

Homomorphic properties

A notable feature of the Paillier cryptosystem is its homomorphic properties along with its non-deterministic encryption (see Electronic voting in Applications for usage). As the encryption function is additively homomorphic, the following identities can be described:

- **Homomorphic addition of plaintexts**

The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts,

$$D(E(m_1, r_1) \cdot E(m_2, r_2) \bmod n^2) = m_1 + m_2 \bmod n.$$

The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts,

$$D(E(m_1, r_1) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n.$$

Upload the file to Storage Environment

In this module user upload their chipper text file to the storage environment. User has the very small size of chipper text file and which environment their want to store the data. If their retrieve the data to see the original plaintext. Enter the correct key to decrypt the data.

2.2.1 ADVANTAGES:

- It covert very large data as small message.
- This technique very usefull in cloud and lage data storage environment.
- High speed in data retrieval.

VI-CONCLUSION

Homomorphic encryption systems have rapidly evolved in the last couple of years. The proposed scheme is suitable for cloud computing environments since it has small bandwidth, low storage requirement, and supports efficient computing on encrypted data. Our solution provides a trade-off between the size of the transmitted ciphertexts and the conversion costs.In this project a cloud based web application was implemented to encrypt data on the cloud servers using Paillier's homomorphic encryption algorithm. The main advantage of this technique is that operations can be performed on encrypted data stored on the cloud without the need for decryption. Hence, user's data is kept confidentiality as the cloud server that operates on it does not know what data it operated upon. Also, if the cloud service provider servers are hacked by malicious attackers, the user's data is secured and cannot be misused as it is homomorphic encrypted. Homomorphic encryption in cloud computing is an active area of research considering the dominance of cloud computing in today's market place. Hence more work can be done in future to build more efficient cloud computing applications similar to the one that is implemented as part of this project. Additional functionality to work with different types of input files can be implemented for this application in the future.

REFERENCES

[1] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. (2013). "A quasipolynomial algorithm for discrete logarithm in finite fields of small characteristic." [Online]. Available: http://eprint.iacr.org/2013/400

[2] J. H. Cheon *et al.*, "Batch fully homomorphic encryption over the integers," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 7881, T. Johansson and P. Nguyen, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 315–335.

[3] K.-M. Chung, Y. Kalai, and S. Vadhan, "Improved delegation of computation using fully homomorphic encryption," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 6223, T. Rabin, Ed. Berlin, Germany: Springer-Verlag, 2010, pp. 483–501.

[4] J.-S. Coron, T. Lepoint, and M. Tibouchi, "Scale-invariant fully homomorphic encryption over the integers," in *Public-Key Cryptography* (Lecture Notes in Computer Science), H. Krawczyk, Ed. Berlin, Germany: Springer-Verlag, 2014, pp. 311–328.

[5] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 196, G. R. Blakley and D. Chaum, Eds. Berlin, Germany: Springer-Verlag, 1984, pp. 10–18.

[6] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," in *Proc. IACR Cryptol.*, 2012, p. 144.

[7] C. Gentry, "Fully homomorphic encryption using ideal lattices,"

in *Proc. 41st Annu. ACM Symp. Theory Comput. (STOC)*, 2009, pp. 169–178.

[8] C. Gentry and S. Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," in *Proc. IEEE 52nd Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2011, pp. 107–109.

[9] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 7417, R. Safavi-Naini and R. Canetti, Eds. Berlin, Germany: Springer-Verlag, 2012, pp. 850–867.

[10] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.