Secure Computation on Cloud Storage:
A Homomorphic Approach

Daya Sagar Gupta, Department of Computer Science and Engineering, Indian School of Mines, Dhanbad, India

G. P. Biswas, Department of Computer Science and Engineering, Indian School of Mines, Dhanbad, India

ABSTRACT

This paper describes a way by which computation on cloud storage is securely possible. A user stores their secret (encrypted) files on cloud storage and later on, retrieves an addition of their original files, however, the cloud system cannot decrypt the stored encrypted files by own. In this paper, the authors use the homomorphic property to securely compute the addition of the files. The implementation of their proposed protocol is based on the computation on the basic properties of elliptic curves and bilinear mapping. The security of proposed encryption technique depends on the hardness of elliptic curve operations.

Keywords: Bilinearity, Cloud Security, Elliptic Curve Cryptography, Homomorphism

1. INTRODUCTION

The Public key encryption (PKE) plays an important role to encrypt a secret data. This paper uses a PKE technique to ensure the security of the designed protocol. The proposed protocol is based on the homomorphic encryption. The idea of homomorphic encryption is firstly proposed by Rivest, Adleman & Dertouzous (1978) as a notion of *privacy homomorphism*. A public key encryption technique which includes the homomorphic property: $E(m_1 \, o_M \, m_2) = E(m_1) \, o_C \, E(m_2)$ is termed as homomorphic encryption. In general, a PKE has three algorithms: *keyGen* which generates a pair of key (public key and private key), *encrypt* which encrypts the message using the public key and *decrypt* which decrypts the message using the private key. Homomorphic encryption also includes these three conventional algorithms with the inclusion of an efficient algorithm *evaluate* which takes cipher texts $c_1, c_2, ..., c_n$ and public key as inputs and produces a valid encryption of some function f on messages $m_1, m_2, ..., m_n$ i.e.

$E(f_M(m_1, m_2 ..... m_n)) <---------- f_C(c_1, c_2 ..... c_n)$.

DOI: 10.4018/JCIT.2015070103

The proposed scheme is based on the elliptic curve cryptography. The elliptic curves play a very important role in the field of cryptography. The security of elliptic curve cryptography is much better than that of the RSA cryptosystem. Our scheme deals with the properties of an elliptic curve. The proposed protocol is based on the bilinear property designed for elliptic curves. The elliptic curve cryptography (ECC) depends on the difficulty provided by elliptic curve operations like addition operation of the points on elliptic curves. Elliptic curve cryptography is nothing but a kind of PKE with a pair of keys i.e. secret and public keys. On the bilinear map, the Computing Diffie-Hellman Problem (CDHP) is difficult, but the Decision Diffie-Hellman Problem (DDHP) is easy. Miller (1985) and Kblitz (1987) independently proposed the security of elliptic curve cryptosystem algorithm which is depending on the discrete logarithm problem of elliptic curves.

In this paper, we are going to present a cryptographic technique which is based on the difficulty of elliptic curve Diffie-Hellman problem and Bilinear Diffie-Hellman problem. The main work is done to secure the cloud storage data. To do so, we use the homomorphic encryption technique in our proposed work. To implement the proposed protocol, we use four algorithms*: keyGen, encrypt, decrypt* and *evaluate*. The *keyGen* is used to generate the public parameters and a secret key. The public parameters are publicly known to others, whereas the secret key is secret to the authenticated user. *Encrypt* algorithm encrypts the secret message which is then stored on a remote server called a cloud. This encrypted message is not understandable to the remote server, *i.e.* the message is securely stored in the cloud so that the original message is not visible to the remote server. *Decrypt* algorithm decrypts the encrypted message to get original message. This *decrypt* algorithm is only used by the authenticated user so that the only true user can get the original message. The *evaluate* algorithm is used for a special purpose. This algorithm is used to show the homomorphic property of the proposed protocol. In this paper, user requests to the remote server for the addition of stored messages stored on cloud storage. The cloud system calculates the addition of messages stored and fulfills the request of the client to return the addition calculated. After getting the calculated addition of message from cloud system, the user decrypts the message and finds the addition of the original messages.

1.1. Literature Review

Benaloh (1987) proposed a technique to verify Secret-Ballot election. The Benaloh' scheme uses the homomorphic property in which the result of an election is verified by each user. Naccache & Stern (1998) proposed a PKE technique which describes a cryptosystem which is based on the hardness of computing higher residues modulo a composite integer. The scheme is described by two versions: the first scheme is *deterministic* which is practically oriented and the second scheme is *probabilistic* which uses a homomorphic property to encrypt the message. Okamoto & Uchiyama (1998) proposed a novel and probably secure PKE technique with some interesting point. The technique is a probabilistic encryption scheme which is based on the homomorphic property.

Pascal Paillier (1999) proposed a public-key cryptosystems Based on Composite Degree Residuosity Classes. This paper describes a secure computational problem which is known as the Composite Residuosity Class Problem, and using this hard problem, Paillier proposed three public-key encryption schemes in which two schemes are homomorphic probabilistic encryption schemes computationally comparable to RSA. Damgard & Jurik (2003) proposed a PKE scheme which is derived from the Paillier cryptosystem. This scheme copies the attractive properties of homomorphism from Paillier encryption. They also found two new interesting properties other than Paillier properties: the first property describes that each and every participating user can deal with the same modulus which is used to generate a public key as well as a private key. In