Homomorphic Encryption for Data Security

Akhilesh Joshi , Mangesh Wagle ,
Nachiket Dhamankar , Nikhil Nandavadekar
Department of Computer Science and Engineering,
Gogte Institute of Technology, Belagavi

1 Introduction

A homomorphic encryption technique allows user to operate ciphertext directly. When user decrypts the resultant cipher, it is same as if operations are carried out on plaintext. Thus, making use of homomorphic encryption assures customers that their data is secure in all state: storage, transmission and processing. Suppose we want to add two numbers 10 and 15, stored in encrypted form (assume 10 as 100 and 15 as 150). The cloud server adds two numbers and store sum as 250, that user decrypts it to the final answer 25.

2 Literature

With rapid proliferation of data over Internet, security became major issue that garnered attention of researchers from academic as well as industry. Traditional standard encryption methods provide security to data in storage state and transmission state. But in processing state, performing operations on data require decryption of data. At this state data is available to cloud provider. Traditional encryption methods are not sufficient to secure data completely. Homomorphic encryption allows user to operate encrypted data directly without decryption.

3 Objectives

Standard encryption techniques used for encryption require decryption to perform operations on data. Decrypting data at cloud data server makes data available to cloud provider. Homomorphic encryption allows users to operate directly on encrypted data.

4 Methodology/ Experimentation

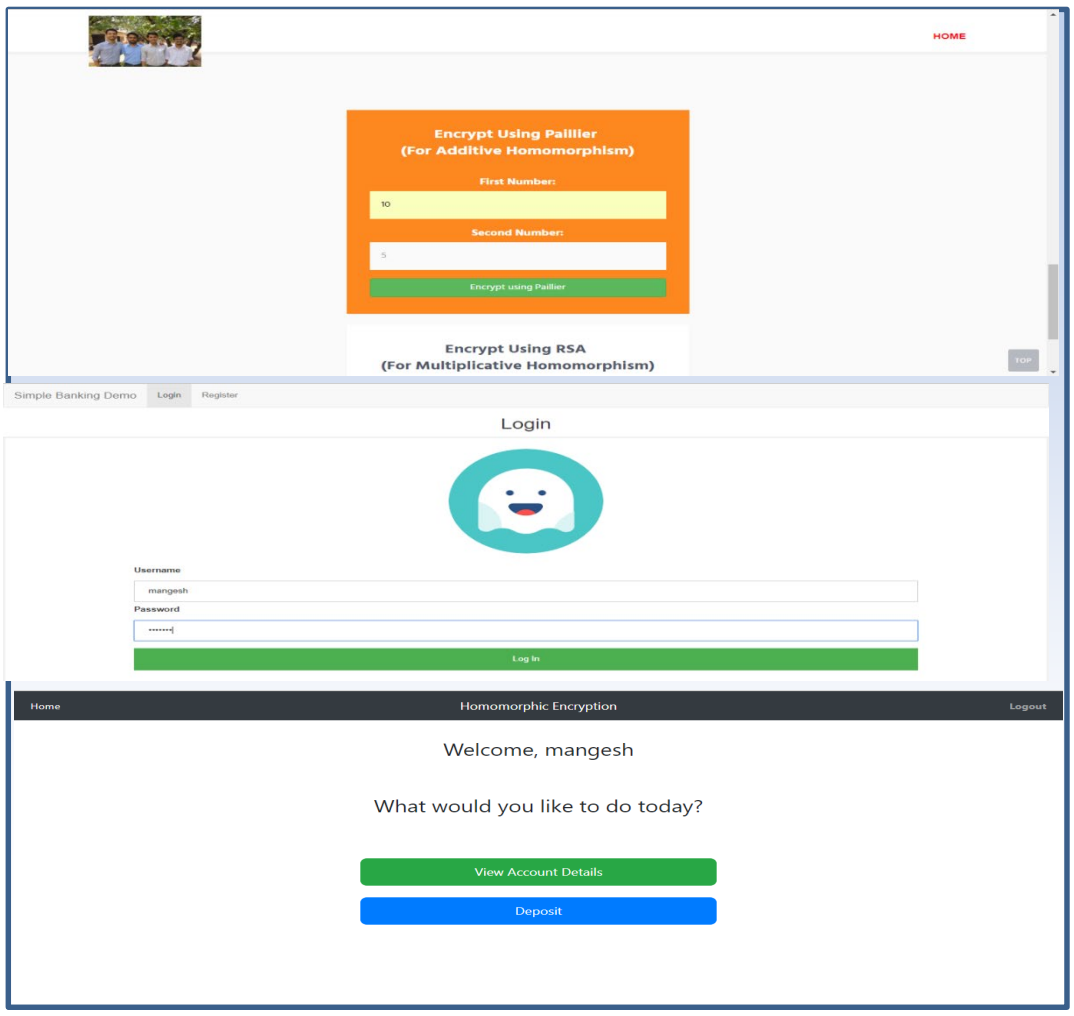
RSA

Key Generation: KeyGen(p,q)	
Input: $p,q \in \mathbb{P}$	
Compute	$n = p \cdot q$ $\varphi(n) = (p-1)(q-1)$
Choose e such that Determine d such that	$\gcd(e, \varphi(n)) = 1$ $e \cdot d \equiv 1 \bmod \varphi(n)$
Output: (pk, sk) public key: $pk = (e, n)$ secret key: $sk = (d)$	
Encryption: Enc(m, pk)	
Input: $m \in \mathbb{Z}_n$	
Compute	$c = m^e \bmod n$
Output: $c \in \mathbb{Z}_n$	
Decryption: Dec(c, sk)	
Input: $c \in \mathbb{Z}_n$	
Compute	$m = c^d \bmod n$
Output: $m \in \mathbb{Z}_n$	

Paillier

Key Generation: KeyGen(p,q)	
Input: $p,q \in \mathbb{P}$	
Compute Choose $g \in \mathbb{Z}^*_{n^2}$ such that	$n = pq$ $\gcd(L(g^\lambda \bmod n^2), n) = 1$ with $L(u) = \frac{u-1}{n}$
Output: (pk, sk) public key: $pk = (n, g)$ secret key: $sk = (p, q)$	
Encryption: Enc(m, pk)	
Input: $m \in \mathbb{Z}_n$	
Choose Compute	$r \in \mathbb{Z}^*_n$ $c = g^m \cdot r^n \bmod n^2$
Output: $c \in \mathbb{Z}_{n^2}$	
Decryption: Dec(c, sk)	
Input: $c \in \mathbb{Z}_{n^2}$	
Compute	$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$
Output: $m \in \mathbb{Z}_n$	

5 Results and Conclusion



6 References

- [1] Maha TEBAA, Said EL HAJII - "Secure Cloud Computing through Homomorphic Encryption"
- [2] ABBAS ACAR, HIDAYET AKSU, and A. SELCUK ULUAGAC - "A Survey on Homomorphic Encryption Schemes: Theory and Implementation"
- [3] Mitchell Harper – "Fully Homomorphic Encryption"
- [4] Mr. Manish M Potey, Dr C A Dhote, Mr Deepak H Sharma – " Homomorphic Encryption for Security of Cloud Data"
- [5] Monique Ogburn, Claude Turner, Pushkar Dahal – "Homomorphic Encryption"
- [6] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan – "Fully Homomorphic Encryption over the Integers"
- [7] N. P. SmartF. Vercauteren – "Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes"
- [8] Craig Gentry, Shai Halevi – "Implementing Gentry's Fully-Homomorphic Encryption Scheme"
- [9] Kornai Kumar Chauhani, Amit K.S. Sanger, Ajai Verma – "Homomorphic Encryption for Data Security in Cloud Computing"