Efficiently Outsourcing multiparty Computation Using Homomorphic Encryption.

1 SUGANYA.R, 2THARANYA.M, 3 KEERTHANA.S, 4INDHU MADHI
123Final Year, 4Assistant Professor, Department of Information Technology.

ABSTRACT- Secure Multiparty Computation (SMC) enables functionalities on their respective original data while keeping these data encrypted throughout the computation. A set of clients to evaluate certain computation. Out sourcing these computations to an untrusted server is desirable, We propose that avoids all these drawbacks: it requires no user interaction (except for data up-download), and it allows evaluating any dynamically chosen function on inputs encrypted under independent public keys.the computation by cryptographic protocol. This protocol is secure in the semi-honest model. Our result in two real-world domains: Privacy-Preserving Face Recognition and Private Smart Metering.

I.INTRODUCTION

Online communication in society is mostly being done through central web-servers it’s process large amounts of private data. Examples of online services are social networks, online auctions, and cloud services to name just a few. In many concerns have been raised regarding data privacy, and serious privacy breaches have occurred [2, 4, 6]. It deal with these privacy threats to sensitive data, the concept of Secure Multiparty Computation (SMC) increasing importance the computation is carried out interactively between more participating parties, in a way that sensitive data is hidden (for example shared among protocol participants) and only the desired output of the computation is available. we focus on solutions based on homomorphic encryption [37].

SMC solutions heavily rely on interaction, since the basic encryption schemes support only limited homomorphisms. In order to perform more complex operations, decryption steps are needed, which require parties holding a secret key, or a share thereof, to be online.

we are considering the following steps in this paper:

1. A set of mutually distrusting clients each having its own public and private key pair, encrypt data using public keys and store these encryptions on a server .2. The function should be computed by on the clients’ data, while all inputs data and intermediate results are private.3. The fact that users are not always online in practice, the ability to compute these functions without any interaction of the users. In particular, this also concerns the users’ retrieval of results.4. Once online, individual user can
retrieve the result while the server learns nothing at all.

II. RELATED WORK.

A. Notation and Security Model.

Standard notation: We write \( x \) if \( X \) is a random variable \( x \) is to be chosen randomly from \( X \) according to its distribution. In the case where \( X \) is solely a set, \( x \) \( \in \) \( U \) denotes that \( x \) is chosen uniformly at random from \( X \). For an algorithm \( A \) we write \( x \leftarrow A(y) \) if \( A \) outputs \( x \) on fixed input \( y \) according to \( A \)’s distribution.

B. Additively Homomorphic Encryption.

A public-key encryption scheme \( E = (\text{KeyGen}, \text{Enc}, \text{Dec}) \) is said to be additively homomorphic if there is an operation “ \( \cdot \) ” in the encrypted domain (usually this is the multiplication) such that for given ciphertexts \( c \) and \( c_1 \), it holds that \( c \cdot c_1 \) is an encryption of the addition of the underlying plaintexts: \( \text{Decsk}(\text{Encpk}(m_1) \cdot \text{Encpk}(m_2)) = m_1 + m_2 \), (1) where \( pk \) and \( sk \) are the public and secret key, respectively, and \( m_1, m_2 \) are two plaintext messages.

C. Group Homomorphic Encryption.

Public-key group homomorphic encryption roughly can be described as usual public-key encryption where the decryption is a group homomorphism.

Definition

A public-key encryption scheme \( E = (\text{KeyGen}, \text{Enc}, \text{Dec}) \) is called group homomorphic, if for every output \( (pk, sk) \) of \( \text{KeyGen}(\kappa) \), the plaintext space \( P_1 \) (original data) and the ciphertext space \( C_1 \) (encrypted data) are non-trivial groups such that the set of encryptions \( C := \{ \text{Encpk}(m; r) \mid m \in P_1, r \in \text{Rnd} \} \) is a non-trivial subgroup of \( _C \) the decryption \( \text{Decsk} \) is a group homomorphism on \( C \), i.e. \( \text{Decsk}(c \cdot c_) = \text{Decsk}(c) \cdot \text{Decsk}(c_) \), for all \( c, c_\in C \).

D. Fully Homomorphic Encryption

An encryption scheme \( E = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Eval}) \) that is homomorphic for all circuits and compact is called fully homomorphic.

The BCP Cryptosystem

The BCP cryptosystem [15] is an additively homomorphic variant of the El Gamal cryptosystem \( \text{Setup}(\kappa) \): security parameter \( \kappa \), choose a prime RSA-modulus \( N = pq \) (i.e., \( p - 1 \)). The plaintext space is \( ZN = 2p' + 1 \) and \( q = 2q' + 1 \) for distinct primes \( p' \) and \( q' \), respectively) of bitlength \( \kappa \). Pick a random element \( g \in Z* N2 \) of order \( pp'qq' \) such that \( gp'q' \mod N2 = 1 + kN \) for \( k \in [1, N] \).

III. SYSTEM MODEL

A. Intermediate Key Aggregation:

The (encrypted) private data from a user \( U \) is not sent to the server \( S \) directly but goes through a chain of intermediate users, this variant allows for the secure aggregation of intermediate public
keys to S’s encrypted data needs to be done by the protocol Key Pro The chain of intermediate client providing different type of key generation can be possible the model automatically produce different type of key.

B. Disclosure by Clients’ Approval.

Assume that user are not supposed to see the result of S’s computation. However, if all participating users give their approval, S is able to read the result. The users should learn the result only if all participating users get together (and not each user independently as in our original construction), this variant can be run instead protocol TransDec in order to make the decryption interactive between all users. Decryption is done if and only if all users participate in this interacts (again in the semi-honest model).

D. Upload Files.

To upload private data to the server s, a user P first it receive the system’s public parameters PP = (N, k, g) to be able to generate its own pair of public and private keys. After these keys are generated, the user P can encrypt its private data using Enc and upload it together with its public key to the server S. Files are sending another user throws server. Upload datum at that time generates an cipher text key (encrypt key). In master key gives a data sender.

IV.PROPOSED SYSTEM

The system mainly proposed for avoiding all draw backs from existing system. This master key sending for server side they encrypted (ciphertext) format. Datum sending menu contains a master key and upload data.

E. File Downloading.

The initial setup of the BCP cryptosystem and the step where users P1, . . . , Pn store their encrypted private input data m1, . . . , mn on the server S – for these two steps, the security follows from the semantic security of the BCP cryptosystem. If matching a cipher key and master key for database. If matching a key into the database the data are downloaded. Otherwise its goes for view original data pages. Again, the only data sent is new cipher texts of blinded messages and due to the semantic security of the underlying crypto system.

1. We assume the existence of a second untrusted server S1 that acts semi-honestly and that does not collude with any of the other person the semi-honest model in Section 2 and on the use of two non-colluding servers.

2. Fixing the number n initially is for reasons of read only. Our building, this number is allowed to change over time. More importantly, users are able to produce their own pair of public and private keys without communicating to some trusted third person. the system is a dynamic process.

3. After this initial setup, users can use the cryptosystem’s Key Gen (independently of any
further person) to produce their respective pair of public and private keys, and to upload encryptions of their private file to the first server S.

4. Function f1 is to be evaluated on the, say, n inputs m1, . . . ,mn of users P1, . . . , Pn, the server S runs a cryptographic protocol with the second server S1 that consists of only four constructing blocks: Key Product, Add, Multiple and Trans Decode. Key Product transforms all cipher texts to encryptions under a single public key, Add and Mult evaluate addition and multiplication gates on encrypted inputs, respectively, and Trans Dec transforms the encrypted result f(m1,m2,m3, . . . ,mn) back to n encryptions each under a different client’s public key.

5. After all computations are done, each user retrieves the encrypted output of the server S which it decrypts locally with its respective private key in order to get the result f(m1,m2,m3, . . . ,mn).

A. Initialization.
Setup process initializes the crypto system and distributes the system’s public parameters. This setup is run by the second server S1.

B. Data Upload
upload private data to the server S, a user P first needs to receive the system’s public parameters PP = (N, k, g) to be able to produce its own pair of public and private keys. After these keys are produce, the user P can encrypt its private file using Encryption and upload it together with its public key to the server S.

C. Data Retrieval.
Each user Pi, i = 1, . . . , n, can get the output of the computation by first receiving (from S) the encryption of F(m1, m2,m3, . . . ,mn) under its public key pk that has been computed processing the subprot Trans Dec, and then decrypting this cipher text by using its private key.

V.IMPLEMENTATION

A. Intermediate Key Aggregation.
The (encrypted) private data from a user U is not sent to the server S directly but goes through a chain of intermediate users, this variant allows for the secure aggregation of intermediate public keys to U’s encrypted data and hence reduces work that the protocol Key Product Intermedia.

Intermediate Key Aggregation. the (encrypted) private data of a user U participate in our protocol is not sent to the server C directly), goes through other user U1, . . . ,Uℓ (a subset of all userP1, . . . , Pn). The protocol presented here optimizes encryption data and keys live in the server the information saved in own drive. give their approval, S is able to read the output encrypted under a public key pk which arrives at user Ui, i = 1, . . . , ℓ.
C. Interactive Decryption by all Users.

Users should learn the output only if all participating users get together (and not each user independently as in our original construction building), this variant can be run instead of protocol TransDecode in order to make the decryption interactive between all users. Decryption is successful if and only if all users participate in this interaction the semi-honest model.

E. KeyProduct in this scenario:

The KeyProd takes all the participating users’ cipher texts (Ai, Bi) as input and transforms these to encryptions under the product Prod.pk of all user participating public keys. The optimization we aim for does the aggregation of the public keys of the clients U, U1, . . . , Uℓ before the cipher texts end up at the server. More precisely, let (A, B) be a cipher text of a message m encrypted under a public key pk which arrives at user Ui, i = 1, . . . , ℓ.

F. Disclosure by Clients’ Approval.

The user are not allowed to see the actual output!F1(m1, . . . , mn) of the computation done by the server S, but if all user approve it, U should be able to retrieve the output broadcasted their respective value Xi. Since the clients never see the (encrypted) private inputs of other clients, they cannot learn more than the decryption of (A, B) which is the result of the computation.

VI. SECURITY ANALYSIS

which blinds the original data plaintext. the actual computations (i.e., the cryptographic protocol) are performed between servers S and U, there is only the initial setup of the cryptosystem and the step where clients P1, . . . , store their encrypted private inputs m1, . . . , mn on server S1. KeyProduceencryptions of f(m1, m2, m3, . . . , mn) under the users’ public keys (pk1, pk2, pk3, . . . , pkn).

Add.

This algorithm is non-interactive and does not involve the server S1 at all, so we are only concerned about U. For U, however, no information leakage is assured by the semantic security. Add just uses the additively homomorphic the cryptosystem.

Mult.

The data sent is new cipher texts of blinded messages and due to the semantic security of the underlying cryptosystem.

Trans Decryption.

The security argument here is the same as for the protocol KeyProduct. it is possible by using random encryptions due to the semantic security of the BCP cryptosystem scheme.

VII. CONCLUSION

The existence of non-colluding but untrusted servers, we presented an efficiently SMC protocol that requires no interaction of the users at all and allows the evaluation of arbitrary functions on inputs that are encrypted (cipher text) under different independent public keys. The protocol to be secure in the semi-honest model and highlighted its practicability by giving experimental outputs. Two applications1.
Privacy preserving face recognition, private smart metering, underlined the applicability of our construction to the real world. To this end integrate our work into tools for automating SMC.

The security analysis considers the semi-honest model only, meaning that all persons follow the protocol description but try to gather information about other persons’ inputs data, intermediate output, security in this model is proven in the existence

**REFERENCE**

1. Armknecht, F., Katzenbeisser, S., Peter, A.: Group homomorphic encryption: