Poster: How to Homomorphically Compute on Data Encrypted under Different Keys?

Asma Aloufi*§, Peizhao Hu*
*Rochester Institute of Technology, NY, USA
§Taif University, Taif, Saudi Arabia

***Abstract*—Homomorphic encryption (HE) supports computations on encrypted data without decryption. This capability can enable secure computation outsourced to a semi-honest cloud without giving up the privacy of user data. One of the demanding applications in cloud computing is secure collaborative computing in which data is contributed by multiple participants and is encrypted using different keys. Current solutions for homomorphic computation on ciphertexts encrypted under multiple keys are inefficient. More specifically, existing works either require key setup before any computation or incur large ciphertext size (at best, grow linearly to the number of involved keys). In this paper, we give an overview of our new approach that leverages the advantages of threshold and multi-key HE to support computations on ciphertexts under different keys while achieving lower complexity. In our collaborative setting, we reduce the ciphertext size from $(N + 1)$ in the state-of-the-art approaches to 2.**

I. Introduction

Homomorphic encryption (HE) is a promising cryptographic technique that protects data in-transmission, at-rest, and in-use without decryption. These capabilities are important for enabling secure computation on private data in the cloud.

Typically in well-known HE schemes, such as BGV [2], homomorphic computation are performed on data that is encrypted under a single key pair. This leads to a weaker security model because all participants have to share the same key pair; that is, they can see each other's private data. Many cloud computing applications require stronger security, such that private data from different individuals is encrypted under different key pairs. This security model allows a group of users to contribute their encrypted data to a cloud evaluator for joint computations without giving away their privacy. We refer to such setting as *secure collaborative computing*.

Collaborative Machine Learning (ML) is an increasingly important application of secure collaborative computing because of the growing interest in ML as a Service (MLaaS) in the cloud. A set of parties cooperate in training predictive models on their private datasets and perform secure classification for the given client inputs, as illustrated in Fig. 1. Joint datasets and models are more diverse and often contain features that help to achieve better accuracy. Operating on encrypted data helps to prevent incidents such as the Equifax data leak [3].

Computing on ciphertexts that are encrypted under different key pairs can be tricky and inefficient. In threshold HE scheme [4], participants generate a joint key, based on the additive homomorphic property of their individual keys, to encrypt their inputs and compute under this joint key. At the end of the evaluation, the users must cooperate to decrypt the result using a multi-party computation (MPC) protocol. Suppose we have N model owners and P clients, as illustrated in Fig. 1. Before any computation, every client needs to generate a joint key with the group of N model owners in a setup phase; that is, we will need to produce and maintain P joint keys. This also means that each model owner has to provide P copies of the encrypted model to the cloud, one for each distinct joint key.

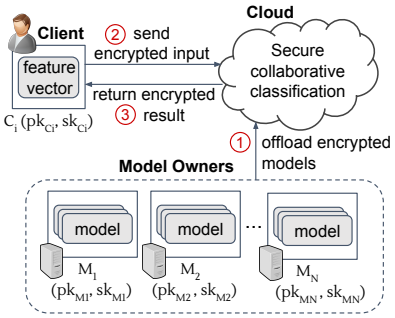
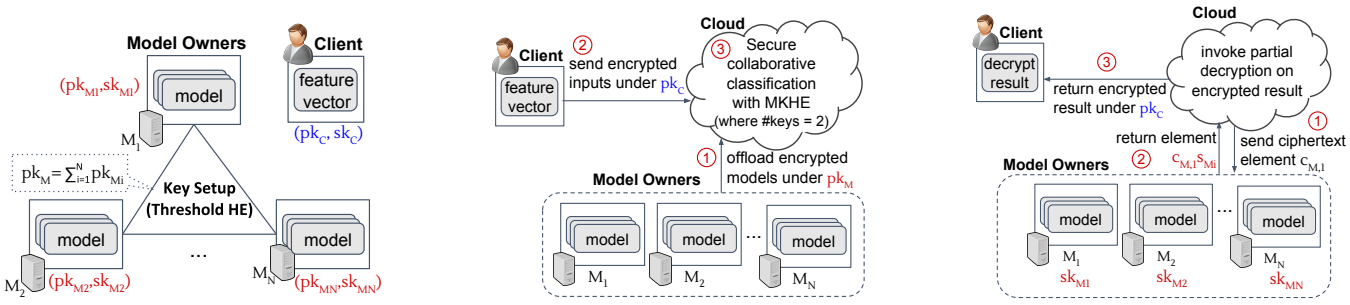


Fig. 1: Secure collaborative machine learning

As an alternative, multi-key HE (MKHE) supports homomorphic computation on ciphertexts encrypted under different keys without a joint key setup. This way, the model owners delegate one encryption copy of their models to the cloud. Ciphertexts can be extended "on-the-fly" to a concatenation of participants' keys at evaluation. The size, more specifically dimension, of an extended ciphertext increases with respect to the number of involved keys. The most efficient construction of MKHE [5] is based on the BGV scheme [2], where the ciphertext size increases linearly. For our collaborative machine learning scenario, an MKHE solution will require each ciphertext to be extended to $N+1$ different keys (i.e., the model owners keys plus the key of the requesting client) before any computation. The efficiency of the system is affected especially if the number of model owners is large because it proportionally increases the ciphertext size.

In this paper, we propose a new approach based on the BGV scheme that supports homomorphic computation over ciphertexts encrypted under multi-keys and produces small

This poster is associated with an SPW paper [1] accepted in the International Workshop on Privacy Engineering (IWPE'19).



(a) Key Setup (b) Encryption and Evaluation (c) Decryption Protocol

Fig. 2: Illustration of the four main phases for secure collaborative evaluation protocol with multi-key support.

ciphertexts. There is no need to set up a joint key for each client with the model owners, and ciphertexts are extended only under *two* different keys instead of $N+1$ keys resulting in a reduction in the ciphertext size.

II. Efficient Secure Computation with Multi-Key

Our new approach leverages both the threshold and multi-key HE techniques. It consists of four different phases: key setup, encryption, evaluation, and decryption. Figure 2 gives an illustration and overview of these four phases.

a) Key setup: In this phase, we generate a key pair $(\mathsf{pk}_{M_i}, \mathsf{sk}_{M_i})$ for each model owner, and a key pair $(\mathsf{pk}_{\mathcal{C}_i}, \mathsf{sk}_{\mathcal{C}_i})$ for each client. Assuming the N model owners do not change frequently, we set up a joint key $\mathsf{pk}_{\mathcal{M}}$ once before the start of the protocol as shown in Fig. 2a. The joint key is generated, in a threshold manner, as the sum of the model owners' individual keys $\mathsf{pk}_{\mathcal{M}} = \sum_{i=1}^{N} = \mathsf{pk}_{M_i}$. It is used to encrypt the models before sending them to the cloud. Note that the encrypted model cannot be decrypted unless all the model owners collaborated since the corresponding secret key $\mathsf{sk}_{\mathcal{M}}$ is shared among them. The joint key can be revoked or updated, but this requires rerunning the threshold key setup and encrypting the models with the new joint key.

b) Encryption: Each model owner M_i encrypts each of their models under the joint key $\mathsf{pk}_{\mathcal{M}}$ as BGV ciphertexts, such that each ciphertext is in the form $c_{\mathcal{M}} = (c_{\mathcal{M},0}, c_{\mathcal{M},1})$. Then, they send the encrypted models to the cloud as shown in step 1 in Fig. 2b. Similarly, the client $\mathcal{C}$ encrypts his inputs under the public key $\mathsf{pk}_{\mathcal{C}}$ as $c_{\mathcal{C}} = (c_{\mathcal{C},0}, c_{\mathcal{C},1})$ and sends the encryptions to the cloud for evaluation (step 2 in Fig. 2b).

c) Evaluation: When the client $\mathcal{C}$ requests an evaluation, the cloud first extends each ciphertext (i.e., encrypted models and client's inputs) under the set of the two keys $\bar{\mathsf{pk}} = \{\mathsf{pk}_{\mathcal{M}}, \mathsf{pk}_{\mathcal{C}}\}$. The extended ciphertext is a concatenated 2 sub-vectors $\bar{c} = (c'_{\mathcal{M}}, c'_{\mathcal{C}})$. For model's ciphertext, $c'_{\mathcal{M}} = c_{\mathcal{M}}$ and $c'_{\mathcal{C}} = 0$. On the other hand, for client's ciphertext, $c'_{\mathcal{M}} = 0$ and $c'_{\mathcal{C}} = c_{\mathcal{C}}$. Then, the cloud performs homomorphic evaluation on extended ciphertexts and outputs a result that is encrypted under the extended key $\bar{\mathsf{pk}}$.

d) Decryption: To decrypt the extended ciphertext result $\bar{c} = (c_{\mathcal{M}}, c_{\mathcal{C}})$, we need the corresponding extended secret key $\bar{\mathsf{sk}} = \{\mathsf{sk}_{\mathcal{M}}, \mathsf{sk}_{\mathcal{C}}\}$ to obtain $\langle \bar{c}, \bar{\mathsf{sk}} \rangle = \langle c_{\mathcal{M}}, \mathsf{sk}_{\mathcal{M}} \rangle + \langle c_{\mathcal{C}}, \mathsf{sk}_{\mathcal{C}} \rangle$. Note that the secret key $\mathsf{sk}_{\mathcal{M}} = (1, s_{\mathcal{M}})$ is secretly shared among N model owners. The client will not be able to decrypt the result without model owners' secret shares of the key. Hence, the cloud invokes an MPC decryption protocol (Fig. 2c), where each model owner computes the decryption component $\rho_i = c_{\mathcal{M},1} s_{M_i}$. Then, the cloud returns the encrypted result and the aggregated component $\sum_{i=1}^{N} \rho_i = c_{\mathcal{M},1} \sum_{i=1}^{N} s_{M_i}$ to the client, who use the latter to decrypt.

Our proposed approach is secure against a semi-honest cloud. The cloud evaluates on encrypted models and client inputs, which are protected under the semantic security of the underlying HE scheme. The decryption protocol follows the dishonest-majority assumption where all users are required to participate in decryption to retrieve the final result.

III. Conclusion

We proposed a new approach that combines the threshold and multi-key HE to support collaborative computation on ciphertexts encrypted under different keys. In the collaborative machine learning setting, our proposed approach (i) removes the need of key setup between a client and each of the set of model owners, (ii) reduces the number of encryption of the same model, and (iii) reduces the ciphertext size with dimension reduction from $(N+1)$ to 2.

References

[1] Asma Aloufi and Peizhao Hu. Collaborative homomorphic computation on data encrypted under multiple keys. International Workshop on Privacy Engineering (IWPE), co-located with IEEE S&P'19.

[2] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 309–325. ACM, 2012.

[3] Sara Ashley O'Brien. Equifax data breach: 143 million people could be affected. https://money.cnn.com/2017/09/07/technology/business/equifax-data-breach, Sep 2017.

[4] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *Advances in Cryptology – EUROCRYPT*, pages 483–501. Springer, 2012.

[5] Long Chen, Zhenfeng Zhang, and Xueqing Wang. Batched multi-hop multi-key fhe from ring-lwe with compact ciphertext extension. In *Theory of Cryptography Conference*, pages 597–627. Springer, 2017.