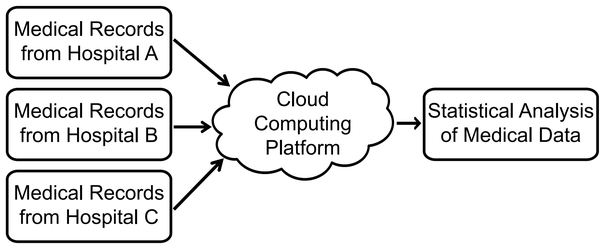
Using Homomorphic Encryption for Large Scale Statistical Analysis

CURIS 2012
David Wu and Jacob Haven
Advised by Professor Dan Boneh

Motivation

Cloud-based solutions have become increasingly popular in the past few years. An example of the cloud-based model is shown below. Here, three different hospitals provide data to the cloud. The cloud computing platform then analyzes and extracts useful information from the data.



One of the main concern with cloud computing has been the privacy and confidentiality of the data. One solution is to send the data encrypted to the cloud. However, we still need to support useful computations on the encrypted data. Fully homomorphic encryption (FHE) is a way of supporting such computations on encrypted data.

We note that while other mechanisms exist for secure computation, they generally require the different data providers to exchange information. Because FHE schemes are *public key* schemes, FHE is much better suited for the scenario where we have many sources of data.

Our Approach

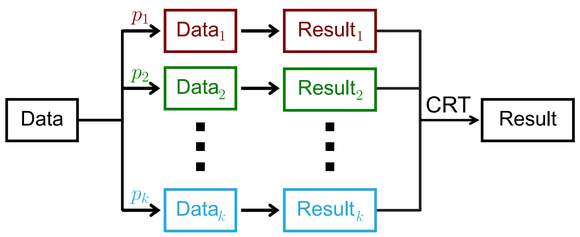
Due to the significant overhead in homomorphic computation, implementations of homomorphic encryption schemes for statistical analysis have been limited to small datasets (≈ 100 data points) and low dimensional data (≈ 2-4 dimensions).

Using recent techniques in batched computation and a different message encoding scheme, we demonstrate the viability of using leveled homomorphic encryption to compute on datasets with over a million elements as well as datasets of much higher dimension.

In particular, we consider two applications of homomorphic encryption: computing the mean and covariance of multivariate data and performing linear regression over encrypted datasets.

Computation over Large Integers

To support computation over large amounts of data, we need to be able to handle large integers (i.e., 128-bit precision). However, it is not computationally feasible to choose message spaces of this magnitude. To support computations with at least 128-bit precision, we leverage the Chinese Remainder Theorem (CRT):

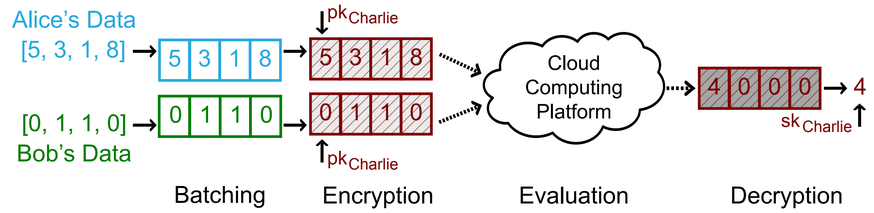


- We choose primes $p_1, p_2, \ldots, p_k$ such that $p_1 p_2 \cdots p_k > 2^{128}$.
- We perform the computation modulo each prime. Given the results of the computation with respect to each prime, we apply the CRT to obtain the value modulo the product of the primes (at least 128-bit precision).
- The computations with respect to each prime is completely independent of the computation with respect to the other primes. As such, all of the computations are naturally parallelizable.

Homomorphic Encryption Scheme (Client Side)

Leveled fully homomorphic encryption (FHE) schemes supports addition and multiplication over ciphertexts. Such schemes are capable of evaluating boolean circuits with bounded depth (determined by the number of multiplications) over ciphertexts, and thus, can perform many computations over the ciphertext.

Consider a scenario where Charlie wants to compute the inner product of Alice's and Bob's data. Note that covariance computation and linear regression can be expressed in terms of matrix products, which can be viewed as a series of inner products.

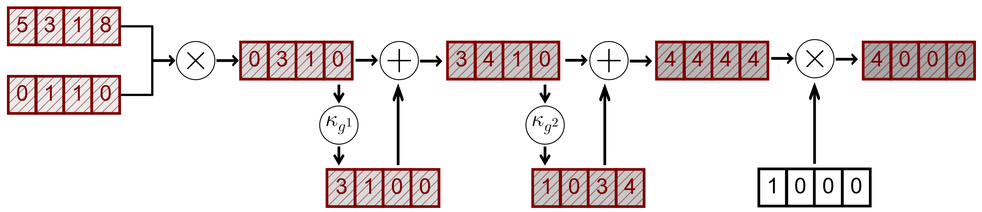


Clear blocks represent plaintext blocks while striped blocks represent ciphertext blocks. The shading in the ciphertext block represents the amount of noise in the ciphertext (explained below).

- **Batching:** Pack multiple plaintext messages (data elements) into a single ciphertext block. This enables the server to evaluate a single instruction on multiple data with low overhead.
- **Encryption:** Encrypt the packed plaintext blocks with the FHE public key (Charlie's public key).
- **Evaluation:** Send the encrypted data to the cloud server for processing.
- **Decryption:** The client (Charlie) decrypts the result using his secret key.

Homomorphic Encryption Scheme (Server Side)

Our leveled FHE scheme supports three basic operations: addition, multiplication, and Frobenius automorphisms. Below, we show how we can use these operations to compute the inner product on encrypted data.



⊕ Element-wise addition of batched ciphertexts.

⊗ Element-wise multiplication of batched ciphertexts.

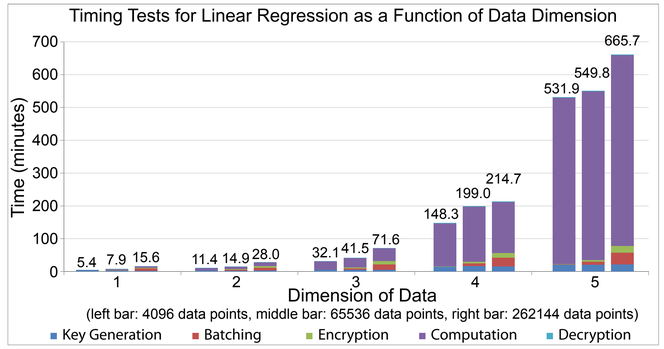
κ_{g^i} Automorphism operation, which can apply arbitrary permutations to elements in the plaintext slots. Used here to rotate slots by i.

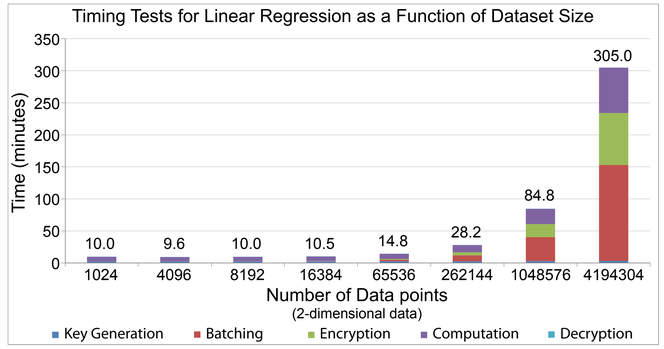
- Because the individual plaintext slots are non-interacting, we use a series of automorphisms to rotate the slots and additions to sum up the entries in a batch
- In the last step of the circuit, we zero out the values of the remaining slots. In the case where the number of slots is not a power of two, this ensures that no additional information about the data is leaked. The result is stored in the first slot of the final ciphertext.
- For security, we must add noise into ciphertexts during the encryption process. Homomorphic operations on ciphertexts increase this noise. In order to decrypt successfully, the noise must be below a chosen threshold.
- In fully homomorphic computation, multiplication is substantially more expensive (both in terms of runtime and amount of noise generated) than addition. We can quantify this by defining the *depth* of a circuit to be the number of multiplications in the circuit. Evaluating deeper circuits requires larger parameters, and correspondingly, longer runtimes. A comparison of addition and multiplication runtimes for different circuit depths is given below:

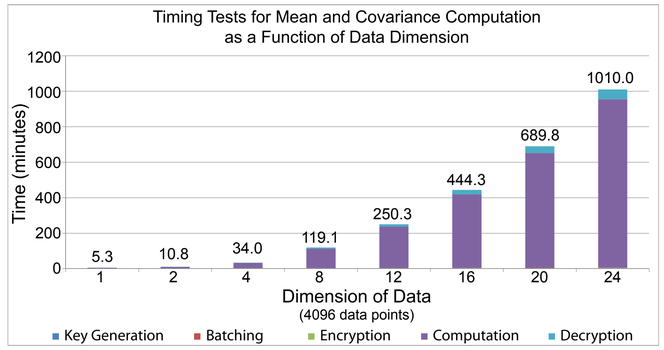
Depth	Time to Perform 1 Addition (**ms**)	Time to Perform 1 Multiplication (**s**)
1	1.94	0.62
2	3.23	2.14
5	10.81	14.11
10	25.44	102.20
20	141.93	771.55

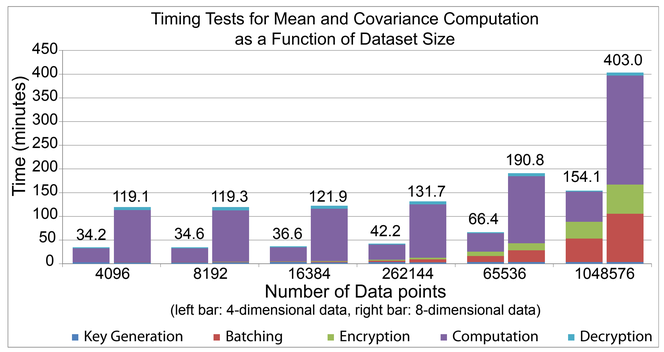
Experiments

Below are results from timing tests illustrating performance of linear regression as well as mean and covariance computation on different datasets. Running times are relative to *one* prime in the CRT decomposition.









Conclusion

- We have constructed a scale-invariant leveled fully homomorphic encryption system.
- Using batching and CRT-based message encoding, we are able to perform *large scale* statistical analysis on millions of data points and data of moderate dimension.